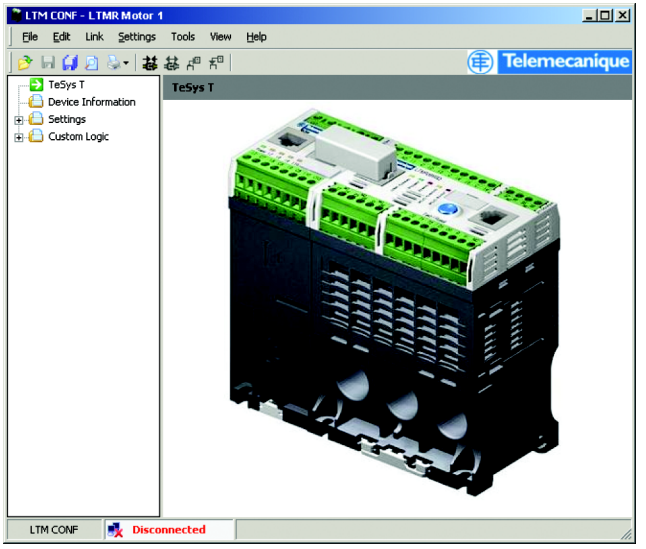
TeSys® T LTM R
Motor Management Controller
Custom Logic Editor
User's Manual

1639507 1.0

12/2006



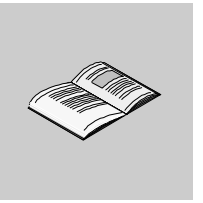
1639507

a brand of Schneider Electric

Telemecanique

www.telemecanique.com

Table of Contents



Safety Information 7

About the Book 9

Chapter 1 Introduction to Custom Logic Editor 11

Introduction to the LTM R Controller 12
Operating Modes 15
Presentation of the Custom Logic Editor 16
Using the Custom Logic Editor 21
Characteristics of the Custom Logic Program 24
Definition of the Custom Logic Variables 25
Definition of the LTM R Variables 26

Chapter 2 Structured Text Language 29

2.1 Creating a Structured Text Program 30
Introducing the Structured Text Editor 31
Structured Text Editor User Interfaces 33
Text View 36
Grid View 40
Logic Commands 43
2.2 Logic Commands 53
Program Logic Commands 54
Boolean Logic Commands 55
Register Logic Commands 69
Timer Logic Commands 85
Latch Logic Commands 89
Counter Logic Commands 91
Math Logic Commands 93
2.3 Structured Text Program Examples 97
How to Check Timers and Multiply Commands 98
How to Create a Truth Table 99
2.4 Compiling and Simulation of a Structured Text Language Program 102
Introduction 103
PCode Window 104
Error Window 106

Output Window 108
Quick Watch Window 110
LTM R Controller Logic Simulator 112

Chapter 3 Function Block Diagram Language 115

3.1 Overview of FBD Language 116
Introduction to the FBD Editor 117
FBD Editor Toolbox 120
3.2 FBD Elements 121
FBD Elements Overview 121
Computation Blocks 122
Inputs Blocks 125
Function Blocks 128
Logic Blocks 133
Outputs Blocks 134
3.3 Programming with the FBD Language 137
Inserting FBD Blocks 138
Creation of Links between Blocks 139
FBD Blocks Properties 141
FBD Resource Management 142
Compiling an FBD program 143
3.4 Manipulating FBD Blocks 144
How to Select Blocks 145
How to Delete and Duplicate Objects 146
3.5 FBD Editor Display Options 147
Other Display Options 148
Workspace Appearance and Graph Options 149

Chapter 4 Connection to the LTM R Controller 153

Hardware Connection 154
Initialization and Connection 156
Transferring Logic Files between the LTM R Controller and Custom Logic Editor 158
Custom Logic Program Transfer and Execution 162

Appendices **163**
Custom Logic Pre-Defined Operating Modes Programs 163

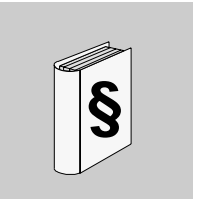
Appendix A LTM R Controller Programming Approach **165**
LTM R Controller Programming Strategy for the 3-Wire Independent Operating Mode . . . 166
3-Wire Independent Mode Programming Example 168
3-Wire Independent Operating Mode Temporary Registers Allocation 170
3-Wire Independent Operating Mode Program Sections 173
3-Wire Independent Operating Mode Allocation Tables 175

Appendix B Pre-Defined Operating Modes Structured Text Programs . . **179**
Structured Text Program for 2-Wire Overload Mode. 180
Structured Text Program for 3-Wire Overload Mode. 191
Structured Text Program for 2-Wire Independent Mode 202
Structured Text Program for 3-Wire Independent Mode 215
Structured Text Program for 2-Wire Reverser Mode. 229
Structured Text Program for 3-Wire Reverser Mode. 245
Structured Text Program for 2-Wire 2-Step Mode. 261
Structured Text Program for 3-Wire 2-Step Mode. 273
Structured Text Program for 2-Wire 2-Speed Mode 285
Structured Text Program for 3-Wire 2-Speed Mode 300

Glossary **317**

Index **321**

Safety Information



Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a Danger or Warning safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates an imminently hazardous situation, which, if not avoided, **will result** in death or serious injury.

⚠ WARNING

WARNING indicates a potentially hazardous situation, which, if not avoided, **can result** in death, serious injury, or equipment damage.

⚠ CAUTION

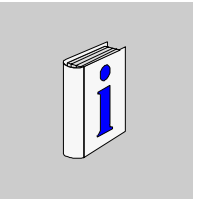
CAUTION indicates a potentially hazardous situation, which, if not avoided, **can result** in injury or equipment damage.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

© 2007 Schneider Electric. All Rights Reserved.

About the Book



At a Glance

Document Scope

This document describes the custom logic editor. It is provided with LTM CONF configuration utility to manage TeSys® T LTM R controllers.

Validity Note

Information in this document only applies to custom logic editor that comes with LTM CONF configuration utility included in the CD.

Related Documents

Title of Documentation	Reference Number
TeSys® T LTM R Modbus User's Manual	1639501
TeSys® T LTM R Profibus User's Manual	1639502
TeSys® T LTM R CANopen User's Manual	1639503
TeSys® T LTM R DeviceNet User's Manual	1639504

You can download this technical publication and other technical information from our website at http://www.telemecanique.com.

Product Related Warnings

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

The application of this product requires expertise in the design and programming of control systems. Only persons with such expertise should be allowed to program and apply this product.
Follow all local and national safety codes and standards.

Failure to follow this instruction can result in death, serious injury, or equipment damage.

User Comments

We welcome your comments about this document. You can reach us by e-mail at techpub@schneider-electric.com

Introduction to Custom Logic Editor

1

At a Glance

Overview

This chapter provides a description of the custom logic editor.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Introduction to the LTM R Controller	12
Operating Modes	15
Presentation of the Custom Logic Editor	16
Using the Custom Logic Editor	21
Characteristics of the Custom Logic Program	24
Definition of the Custom Logic Variables	25
Definition of the LTM R Variables	26

Introduction to the LTM R Controller

Overview

The TeSys® T Motor Management System offers protection, control, and monitoring capabilities for single-phase and 3-phase AC induction motors.

The system is flexible and modular and can be configured to meet the needs of applications in industry. The system is designed to meet the needs for integrated protections systems with open communications and global architecture.

More accurate sensors and solid-state full motor protection ensures better utilization of the motor. Complete monitoring functions enable analysis of motor operating conditions and faster reaction to prevent system downtime.

The system offers diagnostic and statistics functions and configurable warnings and faults, allowing better prediction of component maintenance, and provides data to continuously improve the entire system.

TeSys® T Motor Management System

The two main hardware components of the system are the LTM R controller and the LTM E expansion module. Components such as external motor load current transformers and ground current transformers add additional range to the system. The system can be configured and controlled using either a HMI Operator Terminal, a PLC or a PC. The LTM R PC configuration utility, LTM CONF is required to configure the LTM R controller with custom logic programs.

LTM R controller

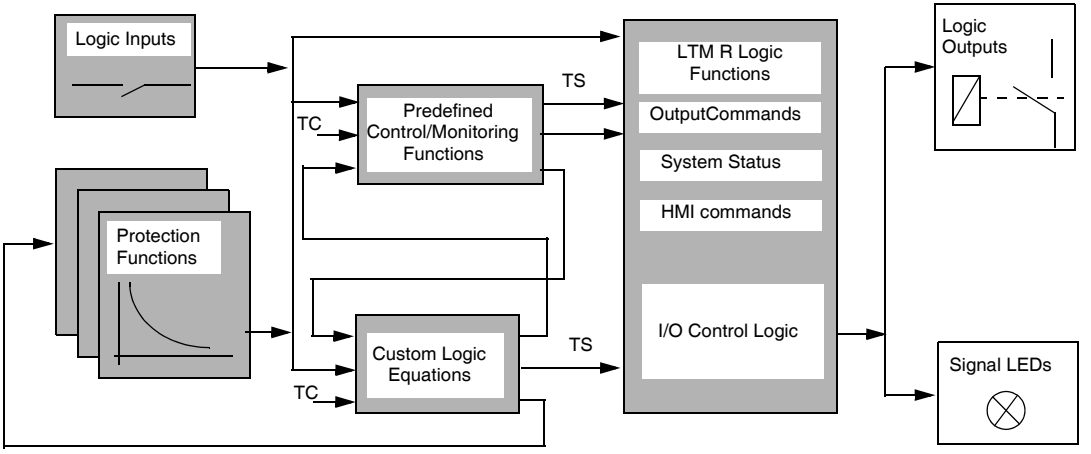
The microprocessor-based LTM R controller is the central component in the system that manages the control, protection and monitoring functions of single-phase and 3-phase ac induction motors.

- The pre-defined functions are those which fit the applications most frequently used in motor starter applications. They are ready to use and are implemented by simple parameter setting after the LTM R controller has been commissioned.
- The pre-defined control and monitoring functions can be adapted for particular needs. The LTM CONF utility allows a programmer to
  - edit pre-defined control functions,
  - alter the default LTM R controller I/O logic assignments.

The pre-defined control program performs 3 main actions:

- acquisition of input data:
  - the protection function fault and warning state
  - external logic data from logic inputs
  - telecommunication commands (TC) received from the control source
- logic processing by the control or monitoring function
- utilization of the processing results:
  - activation of logic outputs
  - activation of LEDs
  - telecommunication signals (TS) sent via a communications link

The control and monitoring function process is displayed below:



Note: Custom logic equations in the diagram above correspond to the control program which is loaded into the LTM R controller's logic memory.

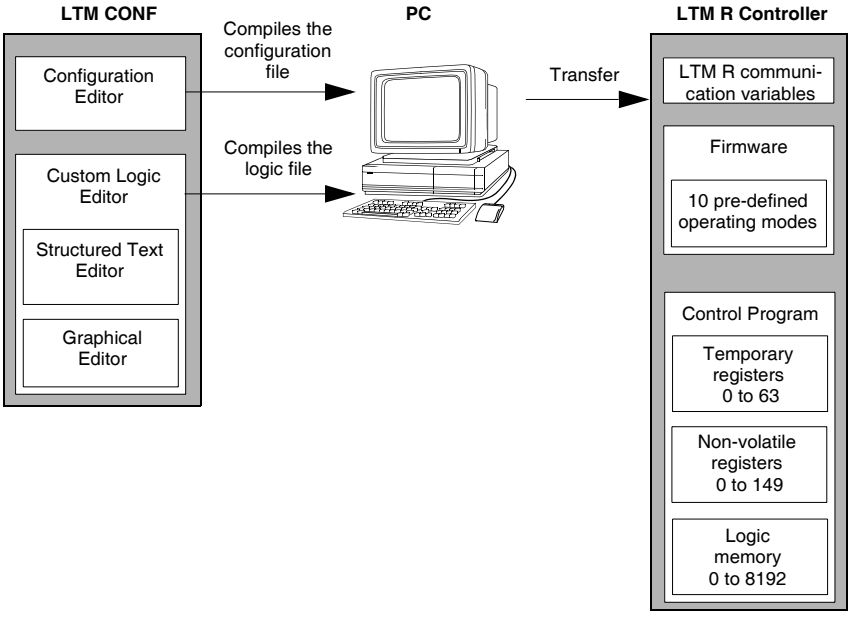
The LTM R controller provides 6 logic inputs, 2 logic outputs, 1 warning relay and 1 fault relay. By adding an expansion module, you can add 4 more logic inputs.

Selecting a pre-defined operating mode automatically assigns the logic inputs to functions and defines the relationship between logic inputs and outputs. Using the custom logic editor, you can change these assignments.

Logic Control Program Actors

Using LTM CONF configuration utility, you have to configure parameters and edit the program file.

The configuration and customization process is displayed below



Note: If one of the pre-defined operating modes is chosen in register 540, the PC transfers settings data to the LTM R controller addresses then the LTM R controller firmware loads the pre-defined operating program from its ROM into the logic memory. If the custom mode is selected, the PC transfers settings data to the LTM R controller registers and the LTM R controller firmware. Then the PC loads the control program from the logic file to the logic memory locations.

Operating Modes

Overview

The LTM R controller supports 10 pre-defined operating modes and 1 custom operating mode (See the section *Motor Control Functions* in the *TeSys T LTM R Motor Management Controller User's Manual*).

Pre-Defined Operating Modes

Each pre-defined operating mode is a list of commands that reside in the read only memory of the LTM R controller.

The LTM R controller uses the commands contained in the pre-defined operating mode control program to:

1. scan the LTM R controller's inputs including configuration parameters, device settings, physical and logic inputs,
2. execute the logic commands,
3. direct the LTM R controller's logical outputs (such as R/W status registers) and physical outputs such as the LEDs.

The installation of the programming software includes 10 pre-defined logic files, one for each combination of:

- Operating Mode (overload, independent, reverser, 2-speed, 2-step), and
- Control Wiring selection: 2-wire (maintained) or 3-wire (impulse).

Custom Operating Mode

The setting for custom operating mode enables you to create your own control program using the custom logic editor in LTM CONF configuration software. To select Custom Operating Mode, navigate to Settings → Motor → Motor Operating Mode, then set the Operating Mode to Custom.

Presentation of the Custom Logic Editor

Overview

A programmable controller reads inputs, solves logic based on a control program, and writes to outputs. You can customize LTM R controller pre-defined control programs using the custom logic editor. The custom logic editor is a powerful programming tool that is only available in LTM CONF configuration utility. Creating a control program for a LTM R controller consists of writing a series of instructions (logic commands) in one of the custom logic programming languages.

Purpose of the Custom Logic Editor

The primary purpose of the custom logic editor is to modify the commands used in the control program that :

- manage local/remote control source
- define LTM R controller I/O logic assignment
- Direct timers such as those used to manage the transitions from low voltage to high voltage contactor in a two/step reduced voltage starter used to implement the start, stop and reset function of a motor controller.
- manage faults
- manage resets

The custom logic editor enables you to modify the behavior of the LTM R controller pre-defined logic programs (operating modes) to meet individual application needs. The custom logic editor allows you to create programs with different types of languages , and then transfer the application to run on a LTM R controller. The modifications may range from minor revisions to complete re-writes of the LTM R controller pre-defined logic programs.

Logic ID

There are 2 ways to create a custom logic program:

- either you edit a pre-defined logic program, what we will call a "custom pre-defined" program,
- or you build a new program from scratch, what we will call a "full custom" program.

The logic ID you must assign to your program has to meet the following rules:

Your program must have a Logic ID in the range...	If it is...
0 to 1	unused
2 to 11	a pre-defined operating mode program(2-wire overload, 3-wire independent, etc.)
12 to 255	a pre-defined program (reserved for future used)
256 to 257	a full custom program
258 to 267	a custom pre-defined program
268 to 511	a custom pre-defined program (reserved for future used)

Custom Pre-Defined Programs

As explained above, a custom pre-defined program is a modified version of a pre-defined operating mode program.

When configured with one of the pre-defined operating modes, the LTM R motor controller manages the control functions using both the firmware in the LTM R controller microprocessor and the PCode.

When configured with a custom pre-defined program, the LTM R controller retains the functions controlled by the LTM R controller microprocessor. Those functions include the following characteristics that are inherent to the "parent' pre-defined operating mode:

- restrictions to what can be written to register 704 (network command register)
- display of the operating state in presentation mode (Fwd/Reverse, Low Speed/ High Speed for example)
- automatic adjustment of power & power factor measurement in 2-step mode with Star-Delta starting selected
- restrictions on which fallback modes may be set through the menus
- specific behaviors regarding the start cycle in 2-step mode
- restrictions on whether the transition timer may be set through the menus

Other behaviors such as interlocking, bump/bumpless, and direct/indirect transitions are managed solely by the custom logic functions and can be modified in either a custom pre-defined program or a full custom one.

Custom Logic Editor Programming Languages

The custom logic editor provides 2 styles of programming languages:

- Structured text language, which is a list instruction language
- Function Block Diagram (FBD), which is an object-oriented programming language.

Note: A third programming language, a ladder logic language, will be added in a future revision of the configuration utility.

Custom Logic Editor Programming Tools

The custom logic editor includes 2 types of programming editors:

- The structured text editor, which is used to create structured text programs. The structured text editor is also referred to as the Text editor.
- The FBD editor, which is used to create Function Block Diagram (FBD) programs. The FBD editor is also referred to as the graphical editor.

Note: A third programming editor, a ladder logic editor, will be added in a future revision of the configuration utility.

Each programming method will satisfy your programming objectives however, the custom logic editor allows you to choose the style of programming method that you prefer.

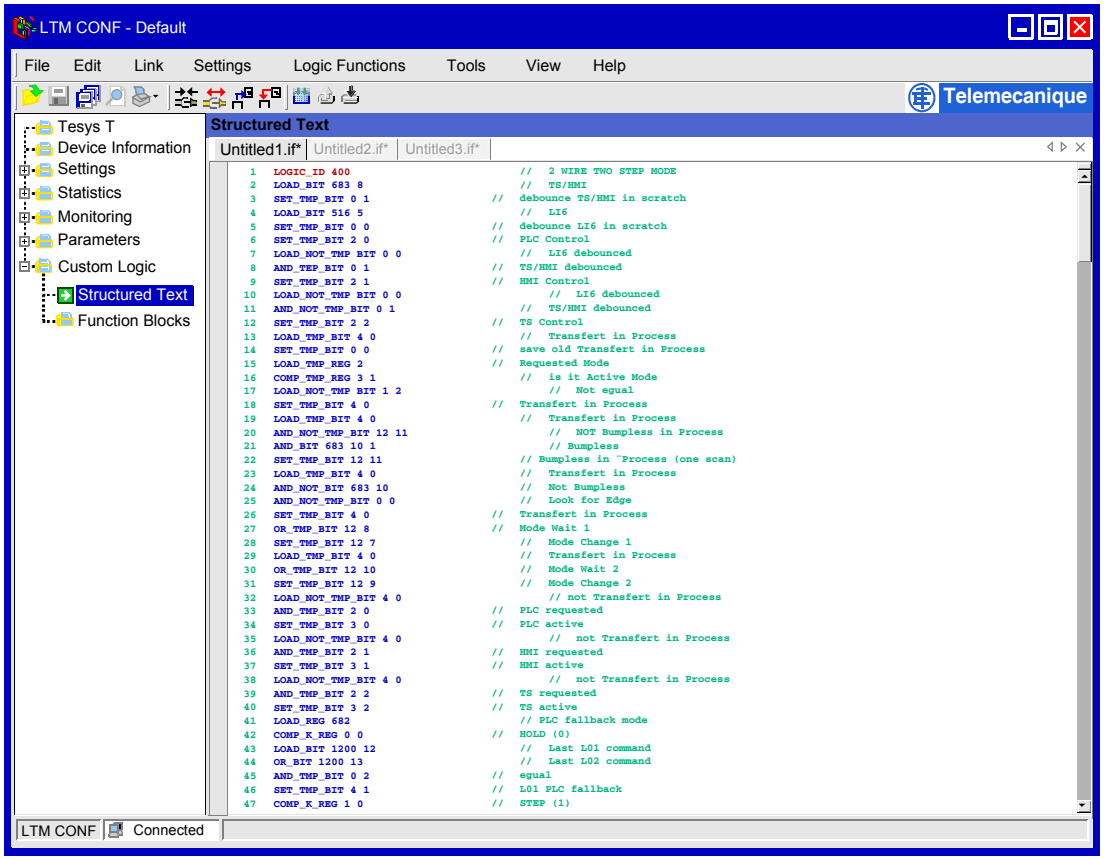
Logic Commands

Both structured text and FBD languages implement the following different types of commands:

- Program logic commands
- Boolean logic commands
- Register logic commands
- Timer logic commands
- Counter logic commands
- Latch logic commands
- Math logic commands

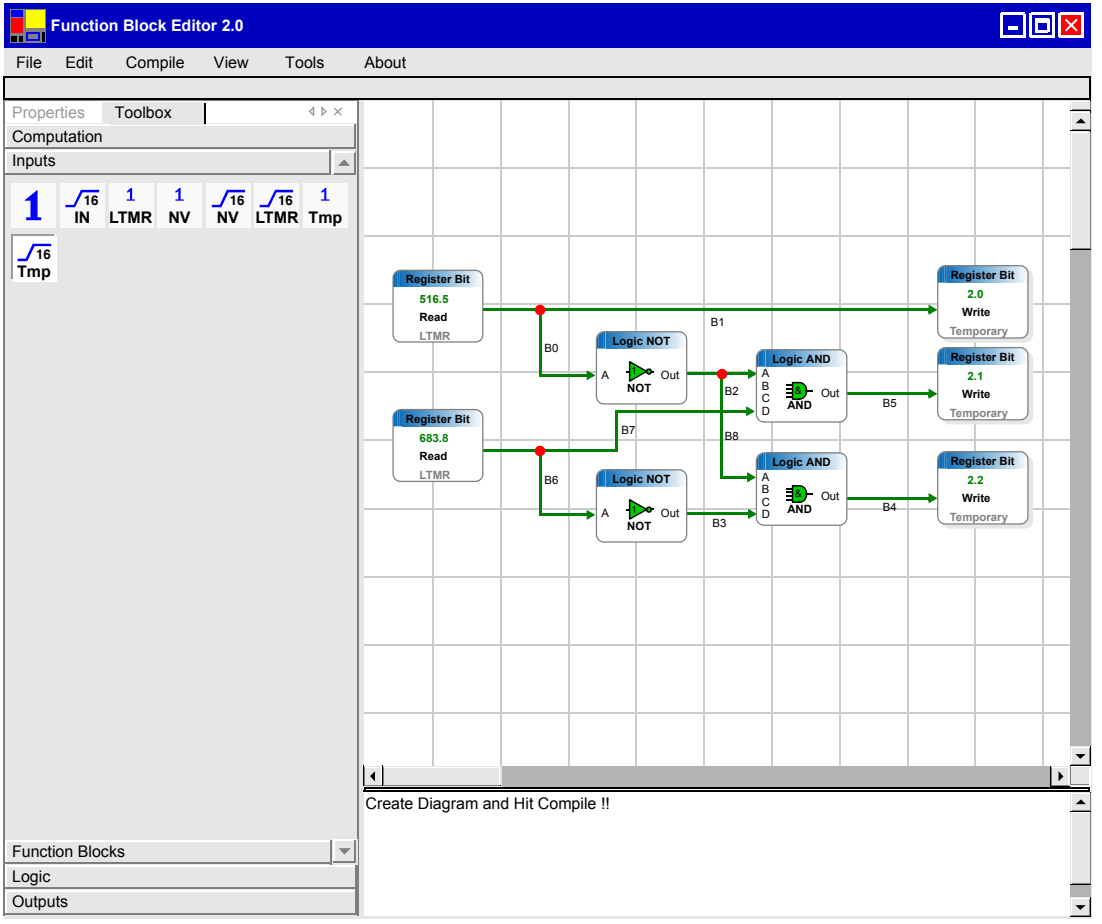
Structured Text Editor

The following illustration shows the structured text editor, integrated in PowerSuite™:


LTM CONF - Default
File
Edit
Link
Settings
Logic Functions
Tools
View
Help
Telemecanique
Tesys T
Device Information
Settings
Statistics
Monitoring
Parameters
Custom Logic
Structured Text
Function Blocks
Structured Text
Untitled1.if*
Untitled2.if*
Untitled3.if*
1 LOGIC_ID 400 // 2 WIRE TWO STEP MODE
2 LOAD_BIT 683 8 // TS/HMI
3 SET_TMP_BIT 0 1 // debounce TS/HMI in scratch
4 LOAD_BIT 516 5 // LI6
5 SET_TMP_BIT 0 0 // debounce LI6 in scratch
6 SET_TMP_BIT 2 0 // PLC Control
7 LOAD_NOT_TMP BIT 0 0 // LI6 debounced
8 AND_TEP_BIT 0 1 // TS/HMI debounced
9 SET_TMP_BIT 2 1 // HMI Control
10 LOAD_NOT_TMP BIT 0 0 // LI6 debounced
11 AND_NOT_TMP_BIT 0 1 // TS/HMI debounced
12 SET_TMP_BIT 2 2 // TS Control
13 LOAD_TMP_BIT 4 0 // Transfert in Process
14 SET_TMP_BIT 0 0 // save old Transfert in Process
15 LOAD_TMP_REG 2 // Requested Mode
16 COMP_TMP_REG 3 1 // is it Active Mode
17 LOAD_NOT_TMP BIT 1 2 // Not egual
18 SET_TMP_BIT 4 0 // Transfert in Process
19 LOAD_TMP_BIT 4 0 // Transfert in Process
20 AND_NOT_TMP_BIT 12 11 // NOT Bumpless in Process
21 AND_BIT 683 10 1 // Bumpless
22 SET_TMP_BIT 12 11 // Bumpless in ¯Process (one scan)
23 LOAD_TMP_BIT 4 0 // Transfert in Process
24 AND_NOT_BIT 683 10 // Not Bumpless
25 AND_NOT_TMP_BIT 0 0 // Look for Edge
26 SET_TMP_BIT 4 0 // Transfert in Process
27 OR_TMP_BIT 12 8 // Mode Wait 1
28 SET_TMP_BIT 12 7 // Mode Change 1
29 LOAD_TMP_BIT 4 0 // Transfert in Process
30 OR_TMP_BIT 12 10 // Mode Wait 2
31 SET_TMP_BIT 12 9 // Mode Change 2
32 LOAD_NOT_TMP_BIT 4 0 // not Transfert in Process
33 AND_TMP_BIT 2 0 // PLC requested
34 SET_TMP_BIT 3 0 // PLC active
35 LOAD_NOT_TMP_BIT 4 0 // not Transfert in Process
36 AND_TMP_BIT 2 1 // HMI requested
37 SET_TMP_BIT 3 1 // HMI active
38 LOAD_NOT_TMP_BIT 4 0 // not Transfert in Process
39 AND_TMP_BIT 2 2 // TS requested
40 SET_TMP_BIT 3 2 // TS active
41 LOAD_REG 682 // PLC fallback mode
42 COMP_K_REG 0 0 // HOLD (0)
43 LOAD_BIT 1200 12 // Last L01 command
44 OR_BIT 1200 13 // Last L02 command
45 AND_TMP_BIT 0 2 // egual
46 SET_TMP_BIT 4 1 // L01 PLC fallback
47 COMP_K_REG 1 0 // STEP (1)
LTM CONF
Connected

FBD Editor

The following illustration shows the FBD editor, integrated in PowerSuite™:



Using the Custom Logic Editor

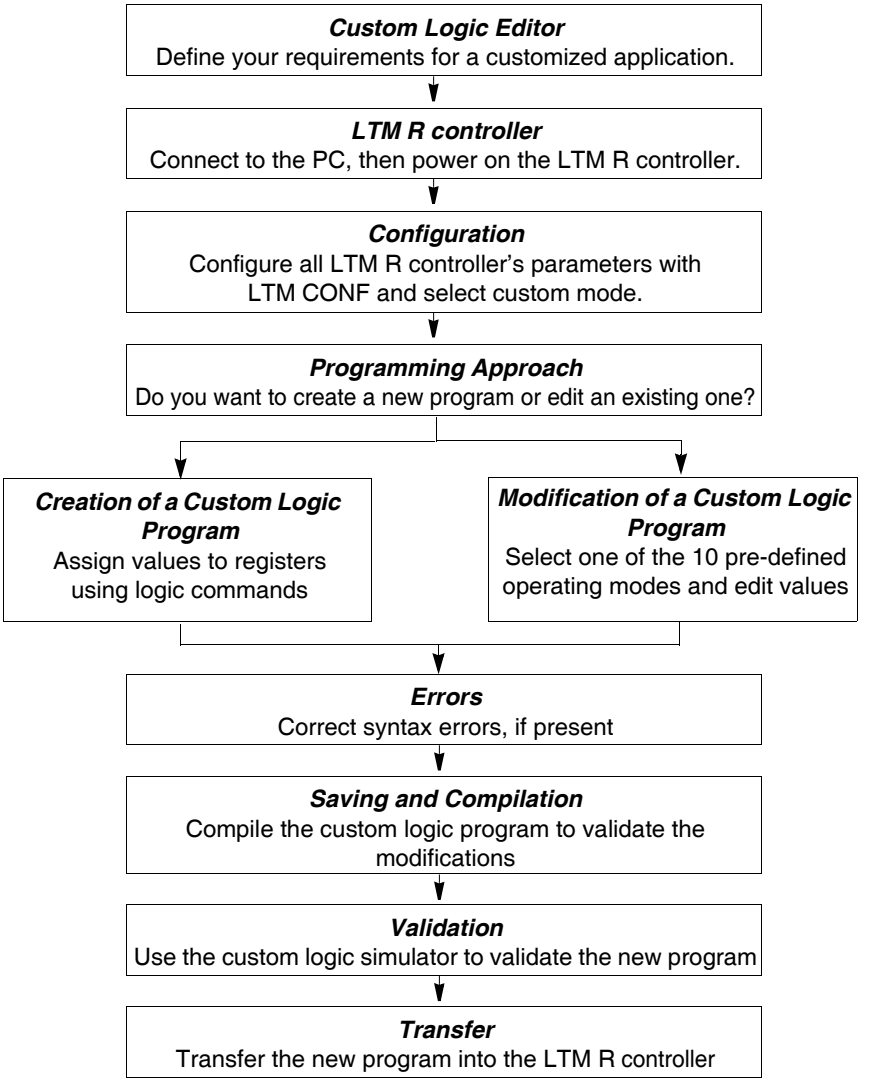
Overview

The custom logic editor enables you to create and validate your own custom logic program to match with your needs. Once it is made, the LTM R controller's firmware loads and execute instructions you created.

Task Flow Diagram

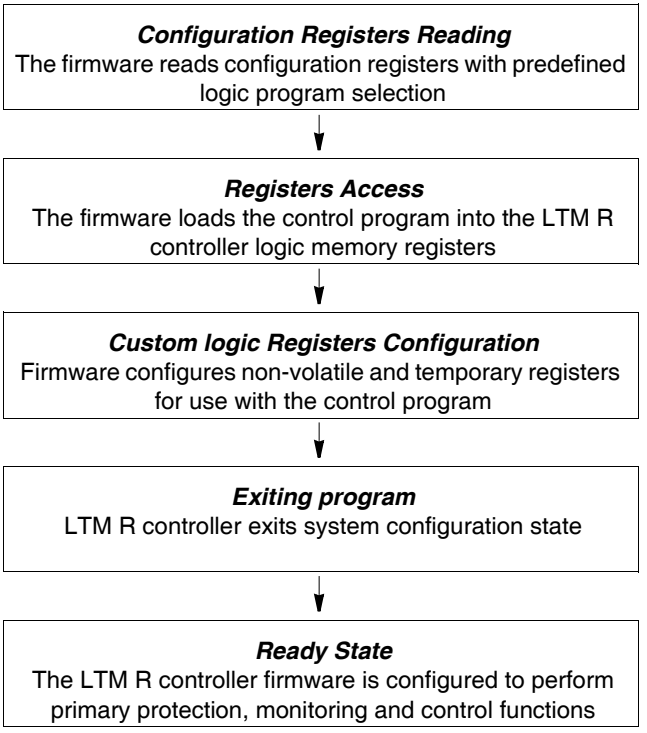
The following diagram shows all of the tasks to be carried out during the creation and modification of a custom logic program (see *LTM R Controller Programming Approach, p. 165*).

Note: The order defined is provided as an example. The order you use will depend on your own work methods.


Custom Logic Editor
Define your requirements for a customized application.
LTM R controller
Connect to the PC, then power on the LTM R controller.
Configuration
Configure all LTM R controller's parameters with LTM CONF and select custom mode.
Programming Approach
Do you want to create a new program or edit an existing one?
Creation of a Custom Logic Program
Assign values to registers using logic commands
Modification of a Custom Logic Program
Select one of the 10 pre-defined operating modes and edit values
Errors
Correct syntax errors, if present
Saving and Compilation
Compile the custom logic program to validate the modifications
Validation
Use the custom logic simulator to validate the new program
Transfer
Transfer the new program into the LTM R controller

Firmware's Task Flow Diagram

The following diagram shows all of the tasks that the firmware will carry out once the custom logic program has been downloaded


Configuration Registers Reading
The firmware reads configuration registers with predefined logic program selection
Registers Access
The firmware loads the control program into the LTM R controller logic memory registers
Custom logic Registers Configuration
Firmware configures non-volatile and temporary registers for use with the control program
Exiting program
LTM R controller exits system configuration state
Ready State
The LTM R controller firmware is configured to perform primary protection, monitoring and control functions

Characteristics of the Custom Logic Program

Introduction

Data transfered to or from the LTM R controller is in the form of 16-bit registers. The registers are numerically ordered and referenced by a 16-bit register address (0 to 65,535).

The custom logic program can modify the values of 3 types of registers:
- Control program registers
- Temporary registers
- Non-volatile registers

Logic Memory Characteristics

The list of commands for the control program is saved in an area of the internal memory of the LTM R controller.

The format of this logic memory is illustrated in the following table:

Memory location	Item	Range	Description
0	Logic Program Size (n)	0 to 8192 0 means a pre-defined operating mode is used	16-bit word
1	Logic Checksum	0 to 65,535	Sum of program memory from offset 2 to n+2
2	Logic Function ID	0 to 255 is reserved for standard LTM R controller modes (e.g. 2=Overload, 4=Independent, etc.) 256 is reserved for full custom logic, not based on a standard LTM R motor controller mode. 257 to 511 is reserved for custom logic based on a standard motor controller mode. (e.g. 258=Custom Overload, 260=Custom Independent, etc.)	Identifier of the custom logic program within the LTM R controller
3	Logic Command/Argument 1	Depending on the logic command (see *p. 53*) type	One word of logic function
4	Logic Command/Argument 2		
5	Logic Command/Argument 3		
...	...	...	...
n+2	Logic Command/Argument n		One word of logic function

Logic Memory Limits

The program size is dependent on the number of logic commands. While in the text editor a command and its arguments will occupy a single line, in the memory, it will occupy as many memory locations as there are arguments.

The command **timer 0.1 980** will occupy 4 memory locations.

Register Locations

The control program, the temporary registers and the non-volatile registers are stored in different areas of the internal memory of the LTM R controller.

Definition of the Custom Logic Variables

Introduction

The custom logic editor enables you to implement commands in the control program which direct the LTM R controller to read or write to the temporary or non-volatile or control program registers.

Temporary Registers

The controller provides registers in temporary memory that can be accessed by logic commands. Because these registers exist in temporary - or volatile - memory, they do not retain their value settings when power to the controller is cycled.

Variables can be stored in temporary registers from 0 to 299. Thus, 300 temporary registers are available.

The LTM R controller defines each custom logic register by an integer describing its address in custom logic memory space. The value of this integer begins at address 0 and the maximum address is equal to 1 less than the number of memory locations available for temporary registers in the LTM R controller. The LTM R controller lists the number of temporary registers available as a value in the LTM R control register 1204, which is the parameter for "custom logic temporary space".

Non-Volatile Registers

TheLTM R controller provides registers in non-volatile memory for use by logic commands. Because these registers exist in non-volatile memory, they retain their value settings when power to the controller is cycled.

Variables can be stored in non-volatile registers from 0 to 63. Thus, 64 non-volatile registers are available.

The LTM R controller defines each custom logic register by an integer describing its address in custom logic memory space. The value of this integer begins at address 0 and the maximum address is equal to 1 less than the number of memory locations available for non-volatile registers in the LTM R controller. The LTM R controller lists the number of non-volatile registers available as a value in the LTM R control register 1205, which is the parameter for "custom logic non-volatile space".

Registers 1301 to 1399

Registers 1301 to 1399 are the General Purpose Registers for logic functions. They are used to exchange information between external sources (such as the PLC) and the custom logic applications.

These registers are read/write and can be edited either by the custom logic functions or via the communication port.

Definition of the LTM R Variables

Overview

Custom logic commands can be used to change the values of read-write data registers of the LTM R controller.

LTM R Variables

Controller memory includes data registers located at addresses ranging from 0 to1399.

Each register is a 16-bit word and is either:
- read-only, with values that cannot be edited, or
- read-write, with values that can be edited.

Accessing Variables

Using the custom logic editor, you can access and edit some of the LTM R controller variables (See the sections on Communication Variables in the *Use* chapter of the *Motor Management Controller :TeSys® T LTM R User's Manual* for a description of these variables and communication protocol registers).

In structured text language, the following logic commands can be used to edit the values of read-write data registers:

Logic Command	Can write to
SET_BIT	1 bit in a read-write data register
SET_NOT_BIT	1 bit in a read-write data register
ON_SET_REG	All bits of a 16-bit read-write data register

Custom Logic Registers

Registers 1200 to 1205 are used by the LTM CONF programming software to access internal register data within the LTM R controller. These registers are also the custom logic registers accessible from the communication ports. These registers are described in the following sections.

The table below lists these registers:

Register	Definition	CONFIG Access	LTM R Access	Range (value)
1200	LTM R logic interface	Read	R/W	0 to 65,535
1201	Logic version			
1202	Logic memory space available			
1203	Logic memory used			
1204	Temporary registers available			
1205	Non-volatile registers available			

Register 1200

Register 1200 is the custom logic interface register. It enables you to configure I/O assignment using the custom logic editor. For example, you can change the reset input from the default I5 to I9.

The following table describes each bit in this register

Bit number	Description
0	Motor Run Command (Default Terminals I1 and I2)
1	Motor Stop Command (Default Terminal I4)
2	Reset Command (Default Terminal I5)
3	Step 2 Active
4	Transition Timer Active
5	Phase Direction Reversed
6	Remote Control (Default Terminal I6)
7	FLA Select (0=FLA1, 1=FLA2)
8	External Fault (0=No Fault, 1=Fault)
9	Aux 1 LED (PowerSuite™ and HMI)
10	Aux 2 LED (PowerSuite™ and HMI)
11	Stop LED (PowerSuite™ and HMI)
12	Logic Output 1 (LTM R)
13	Logic Output 2 (LTM R)
14	Logic Output 3 (LTM R)
15	Logic Output 4 (LTM R)

Register 1201

Register 1201 indicates the custom logic capability version. The version number identifies a specific group logic commands supported by the LTM R controller.

Register 1202

Register 1202 defines the logic memory space available, that is, the number of non-volatile LTM R controller logic memory words (16 bits) available to save logic commands.

Register 1203

Register 1203 defines the logic memory used, that is, the number of non-volatile LTM R logic memory words (16 bits) used by logic commands which are currently stored in the LTM R controller.

Register 1204

Register 1204 defines the number of temporary registers provided by the LTM R controller.

Register 1205

Register 1205 defines the number of non-volatile registers provided by the LTM R controller.

Structured Text Language

2

At a Glance

Overview

The structured text editor enables you to create a custom logic program based on the structured text programming language.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
2.1	Creating a Structured Text Program	30
2.2	Logic Commands	53
2.3	Structured Text Program Examples	97
2.4	Compiling and Simulation of a Structured Text Language Program	102

2.1 Creating a Structured Text Program

At a Glance

Summary

This section describes the creation of a program with the structured text editor.

Use the structured text editor to modify the pre- defined operating program by
- changing the input and output assignments of the logic functions and,
- adding new logic functions that will change the step by step instructions of the original program.

Create a new program by designing the step instructions tailored to the specific requirements of the application.

What's in this Section?

This section contains the following topics:

Topic	Page
Introducing the Structured Text Editor	31
Structured Text Editor User Interfaces	33
Text View	36
Grid View	40
Logic Commands	43

Introducing the Structured Text Editor

Overview

The structured text editor is a feature of TeSys® configuration software. Use the structured text editor to view an existing logic file or to create a new logic file using a instruction based / text language, rather than a graphics based programming language.

Editing a Structured Text Program

The easiest way to create a new logic file is to begin with a logic file for one of the the pre-defined operating modes. Your installation of the custom logic editor comes with 10 pre-defined logic files, one for each combination of:

- operating mode (2-speed, 2-step, independent, overload, reverser), and
- control wiring selection (2-wire, 3-wire).

Each logic file bears a descriptive name (e.g. "3-wire-reverser") and a file extension of ".lf".

How to Edit a Structured Text Program

Follow these steps to create a structured text program from a pre-defined operating mode program:

Step	Action
1	Navigate to Settings → Motor → Motor Operating Mode to use the logic editor in the configuration software.
2	Set the operating mode to **Custom**
3	Open the predefined logic file that is closest to your application's requirements (see *LTM R Controller Programming Approach, p. 165*). Then use the structured text editor's logic commands (see *p. 43*) to alter the default logic file to suit your application's needs.
4	Use the **Compile** command in the logic functions menu or on the icon bar to validate the custom logic file when you finished editing the logic file. **Result**: The PCode (Pseudo Code) window notifies you whether the logic file compiled successfully. The PCode window is a feature of LTM CONF programming software.
5	Download, after the logic file is compiled, the completed logic file from the custom logic editor to the controller using the logic functions menu **Download Program to Device** command.

Custom Logic Editor User Interface

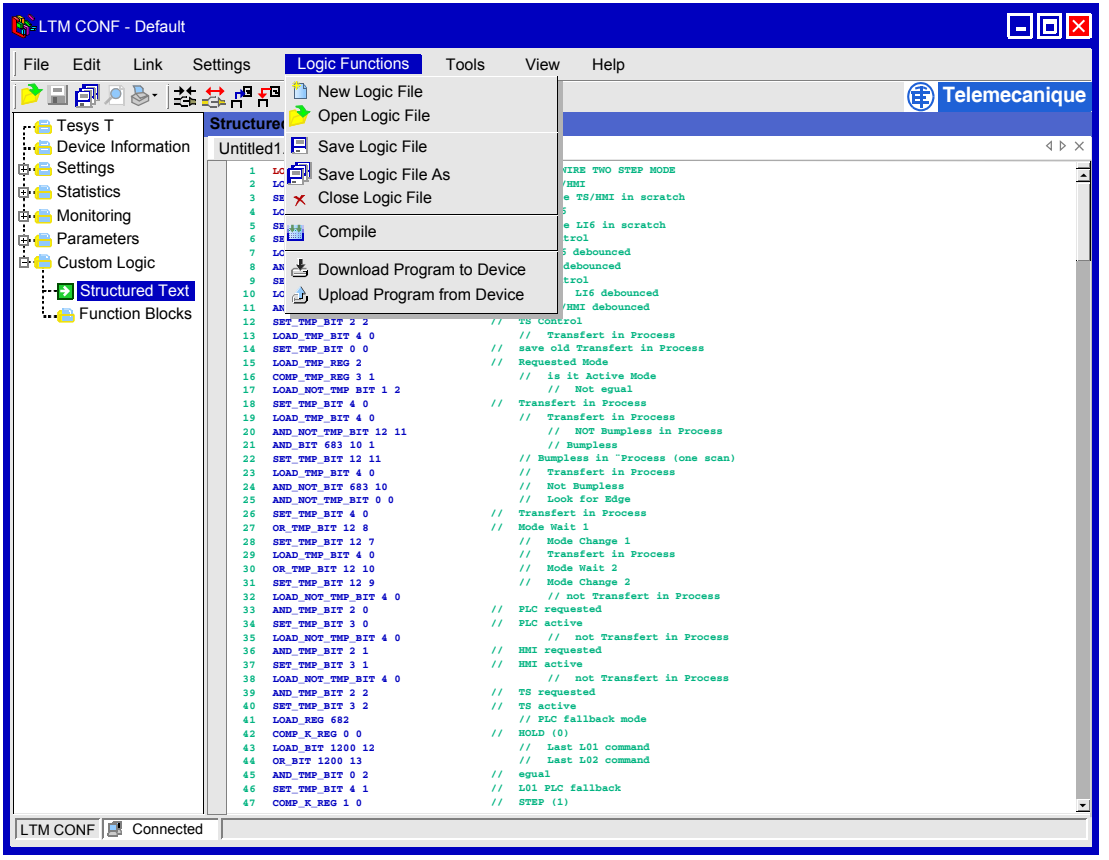
To open the structured text editor, navigate in the configuration software tree (on the left of the screen) to Custom Logic → Structured Text. This will:

- open the structured text editor in the main window, and
- enable the logic functions menu.

The structured text editor is available regardless of whether the configuration software is connected to the controller. However, many of the logic functions menu items will be enabled only when:

- a logic file is open in the structured text editor, and
- the configuration software is connected to the controller.

When a logic file is open and the configuration software is connected to the controller, the structured text editor and logic functions menu look like this:



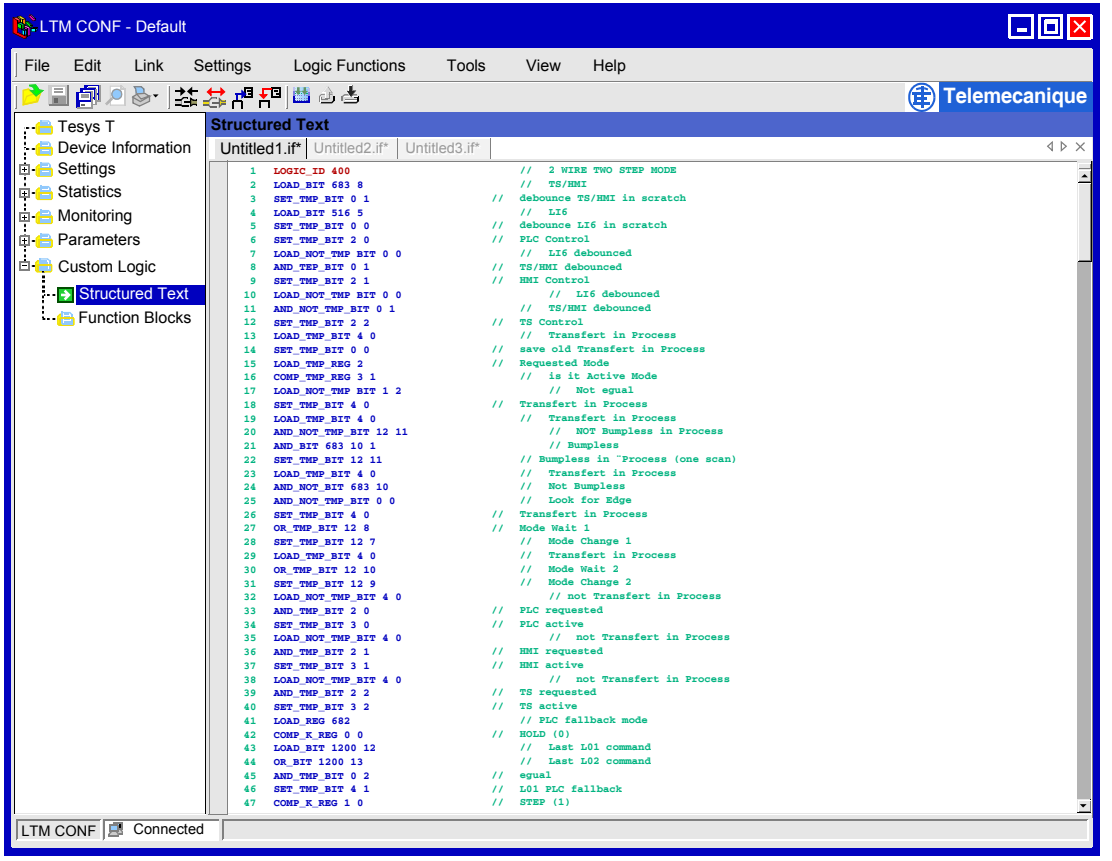
Structured Text Editor User Interfaces

overview

There are two different ways to create a program with the structured text editor. You can choose to use either the Text or the Grid view.

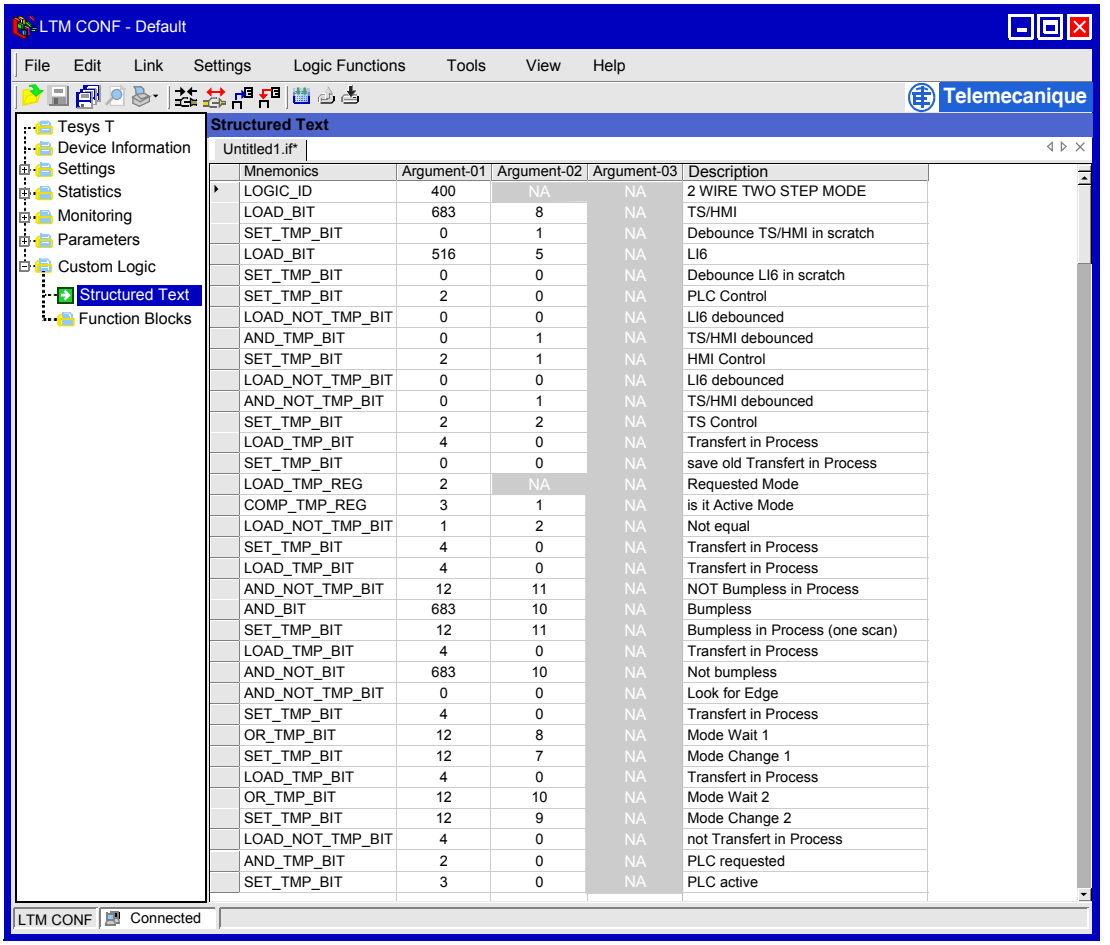
Text View

The following illustration shows the structured text editor in Text view:


LTM CONF - Default
File Edit Link Settings Logic Functions Tools View Help
Telemecanique
Tesys T
Device Information
Settings
Statistics
Monitoring
Parameters
Custom Logic
Structured Text
Function Blocks
Structured Text
Untitled1.if*
Untitled2.if*
Untitled3.if*
1 LOGIC_ID 400 // 2 WIRE TWO STEP MODE
2 LOAD_BIT 683 8 // TS/HMI
3 SET_TMP_BIT 0 1 // debounce TS/HMI in scratch
4 LOAD_BIT 516 5 // LI6
5 SET_TMP_BIT 0 0 // debounce LI6 in scratch
6 SET_TMP_BIT 2 0 // PLC Control
7 LOAD_NOT_TMP BIT 0 0 // LI6 debounced
8 AND_TEP_BIT 0 1 // TS/HMI debounced
9 SET_TMP_BIT 2 1 // HMI Control
10 LOAD_NOT_TMP BIT 0 0 // LI6 debounced
11 AND_NOT_TMP_BIT 0 1 // TS/HMI debounced
12 SET_TMP_BIT 2 2 // TS Control
13 LOAD_TMP_BIT 4 0 // Transfert in Process
14 SET_TMP_BIT 0 0 // save old Transfert in Process
15 LOAD_TMP_REG 2 // Requested Mode
16 COMP_TMP_REG 3 1 // is it Active Mode
17 LOAD_NOT_TMP BIT 1 2 // Not egual
18 SET_TMP_BIT 4 0 // Transfert in Process
19 LOAD_TMP_BIT 4 0 // Transfert in Process
20 AND_NOT_TMP_BIT 12 11 // NOT Bumpless in Process
21 AND_BIT 683 10 1 // Bumpless
22 SET_TMP_BIT 12 11 // Bumpless in "Process (one scan)
23 LOAD_TMP_BIT 4 0 // Transfert in Process
24 AND_NOT_BIT 683 10 // Not Bumpless
25 AND_NOT_TMP_BIT 0 0 // Look for Edge
26 SET_TMP_BIT 4 0 // Transfert in Process
27 OR_TMP_BIT 12 8 // Mode Wait 1
28 SET_TMP_BIT 12 7 // Mode Change 1
29 LOAD_TMP_BIT 4 0 // Transfert in Process
30 OR_TMP_BIT 12 10 // Mode Wait 2
31 SET_TMP_BIT 12 9 // Mode Change 2
32 LOAD_NOT_TMP_BIT 4 0 // not Transfert in Process
33 AND_TMP_BIT 2 0 // PLC requested
34 SET_TMP_BIT 3 0 // PLC active
35 LOAD_NOT_TMP_BIT 4 0 // not Transfert in Process
36 AND_TMP_BIT 2 1 // HMI requested
37 SET_TMP_BIT 3 1 // HMI active
38 LOAD_NOT_TMP_BIT 4 0 // not Transfert in Process
39 AND_TMP_BIT 2 2 // TS requested
40 SET_TMP_BIT 3 2 // TS active
41 LOAD_REG 682 // PLC fallback mode
42 COMP_K_REG 0 0 // HOLD (0)
43 LOAD_BIT 1200 12 // Last L01 command
44 OR_BIT 1200 13 // Last L02 command
45 AND_TMP_BIT 0 2 // egual
46 SET_TMP_BIT 4 1 // L01 PLC fallback
47 COMP_K_REG 1 0 // STEP (1)
LTM CONF
Connected

Grid View

The following illustration shows the structured text editor in Grid view:



Mnemonics	Argument-01	Argument-02	Argument-03	Description
LOGIC_ID	400	NA	NA	2 WIRE TWO STEP MODE
LOAD_BIT	683	8	NA	TS/HMI
SET_TMP_BIT	0	1	NA	Debounce TS/HMI in scratch
LOAD_BIT	516	5	NA	LI6
SET_TMP_BIT	0	0	NA	Debounce LI6 in scratch
SET_TMP_BIT	2	0	NA	PLC Control
LOAD_NOT_TMP_BIT	0	0	NA	LI6 debounced
AND_TMP_BIT	0	1	NA	TS/HMI debounced
SET_TMP_BIT	2	1	NA	HMI Control
LOAD_NOT_TMP_BIT	0	0	NA	LI6 debounced
AND_NOT_TMP_BIT	0	1	NA	TS/HMI debounced
SET_TMP_BIT	2	2	NA	TS Control
LOAD_TMP_BIT	4	0	NA	Transfert in Process
SET_TMP_BIT	0	0	NA	save old Transfert in Process
LOAD_TMP_REG	2	NA	NA	Requested Mode
COMP_TMP_REG	3	1	NA	is it Active Mode
LOAD_NOT_TMP_BIT	1	2	NA	Not equal
SET_TMP_BIT	4	0	NA	Transfert in Process
LOAD_TMP_BIT	4	0	NA	Transfert in Process
AND_NOT_TMP_BIT	12	11	NA	NOT Bumpless in Process
AND_BIT	683	10	NA	Bumpless
SET_TMP_BIT	12	11	NA	Bumpless in Process (one scan)
LOAD_TMP_BIT	4	0	NA	Transfert in Process
AND_NOT_BIT	683	10	NA	Not bumpless
AND_NOT_TMP_BIT	0	0	NA	Look for Edge
SET_TMP_BIT	4	0	NA	Transfert in Process
OR_TMP_BIT	12	8	NA	Mode Wait 1
SET_TMP_BIT	12	7	NA	Mode Change 1
LOAD_TMP_BIT	4	0	NA	Transfert in Process
OR_TMP_BIT	12	10	NA	Mode Wait 2
SET_TMP_BIT	12	9	NA	Mode Change 2
LOAD_NOT_TMP_BIT	4	0	NA	not Transfert in Process
AND_TMP_BIT	2	0	NA	PLC requested
SET_TMP_BIT	3	0	NA	PLC active

Default Settings

When you choose to open a logic file, or to create a new one, the logic file will always open in Text view.

Switching from Text View to Grid View

There are two ways to change between Text view and Grid view:

- In the upper menu bar, click on View and choose Text view or Grid view, or
- right-click on the file name and choose a view.

Editing several programs

You can create or modify several custom logic programs at the same time. Just click on the file name to switch between them.

For instance, in the Text view above, click either untitled1.lf, untitled2.lf, or untitled3.lf, depending on the program you wish to edit.

Text View

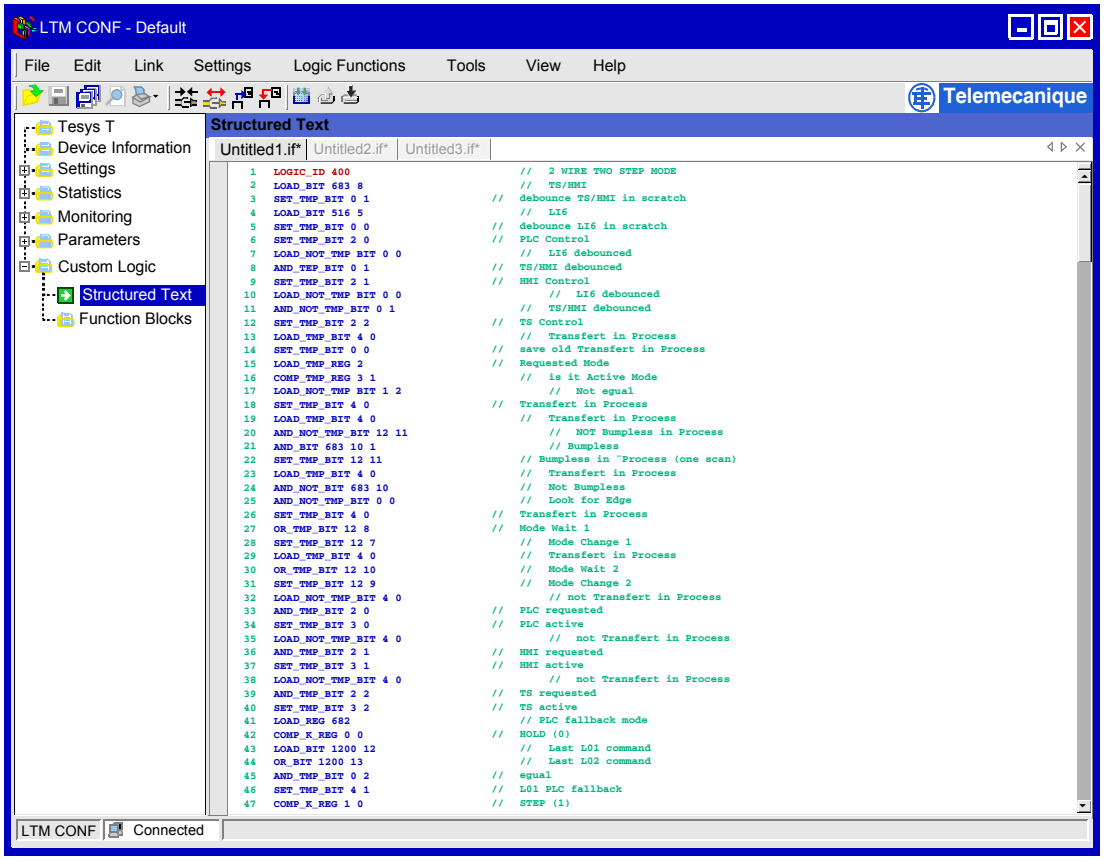
Introduction

A program written in list language consists of a series of instructions executed sequentially by the controller. Each list instruction is represented by a single program line and consists of three components:

- Line number
- Logic command (Mnemonics)
- Argument(s)

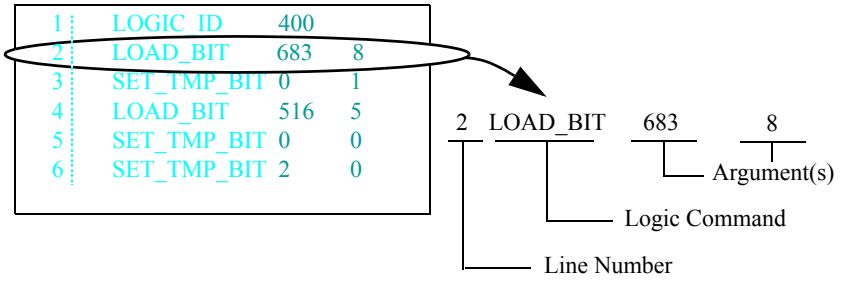
Example of a Structured Text Program in Text View

The following is an example of a program created with the structured text editor in Text view.


LTM CONF - Default
File Edit Link Settings Logic Functions Tools View Help
Telemecanique
Tesys T
Device Information
Settings
Statistics
Monitoring
Parameters
Custom Logic
Structured Text
Function Blocks
Structured Text
Untitled1.if*
Untitled2.if*
Untitled3.if*
1 LOGIC_ID 400 // 2 WIRE TWO STEP MODE
2 LOAD_BIT 683 8 // TS/HMI
3 SET_TMP_BIT 0 1 // debounce TS/HMI in scratch
4 LOAD_BIT 516 5 // LI6
5 SET_TMP_BIT 0 0 // debounce LI6 in scratch
6 SET_TMP_BIT 2 0 // PLC Control
7 LOAD_NOT_TMP_BIT 0 0 // LI6 debounced
8 AND_TEP_BIT 0 1 // TS/HMI debounced
9 SET_TMP_BIT 2 1 // HMI Control
10 LOAD_NOT_TMP_BIT 0 0 // LI6 debounced
11 AND_NOT_TMP_BIT 0 1 // TS/HMI debounced
12 SET_TMP_BIT 2 2 // TS Control
13 LOAD_TMP_BIT 4 0 // Transfert in Process
14 SET_TMP_BIT 0 0 // save old Transfert in Process
15 LOAD_TMP_REG 2 // Requested Mode
16 COMP_TMP_REG 3 1 // is it Active Mode
17 LOAD_NOT_TMP_BIT 1 2 // Not equal
18 SET_TMP_BIT 4 0 // Transfert in Process
19 LOAD_TMP_BIT 4 0 // Transfert in Process
20 AND_NOT_TMP_BIT 12 11 // NOT Bumpless in Process
21 AND_BIT 683 10 1 // Bumpless
22 SET_TMP_BIT 12 11 // Bumpless in ~Process (one scan)
23 LOAD_TMP_BIT 4 0 // Transfert in Process
24 AND_NOT_BIT 683 10 // Not Bumpless
25 AND_NOT_TMP_BIT 0 0 // Look for Edge
26 SET_TMP_BIT 4 0 // Transfert in Process
27 OR_TMP_BIT 12 8 // Mode Wait 1
28 SET_TMP_BIT 12 7 // Mode Change 1
29 LOAD_TMP_BIT 4 0 // Transfert in Process
30 OR_TMP_BIT 12 10 // Mode Wait 2
31 SET_TMP_BIT 12 9 // Mode Change 2
32 LOAD_NOT_TMP_BIT 4 0 // not Transfert in Process
33 AND_TMP_BIT 2 0 // PLC requested
34 SET_TMP_BIT 3 0 // PLC active
35 LOAD_NOT_TMP_BIT 4 0 // not Transfert in Process
36 AND_TMP_BIT 2 1 // HMI requested
37 SET_TMP_BIT 3 1 // HMI active
38 LOAD_NOT_TMP_BIT 4 0 // not Transfert in Process
39 AND_TMP_BIT 2 2 // TS requested
40 SET_TMP_BIT 3 2 // TS active
41 LOAD_REG 682 // PLC fallback mode
42 COMP_K_REG 0 0 // HOLD (0)
43 LOAD_BIT 1200 12 // Last L01 command
44 OR_BIT 1200 13 // Last L02 command
45 AND_TMP_BIT 0 2 // equal
46 SET_TMP_BIT 4 1 // L01 PLC fallback
47 COMP_K_REG 1 0 // STEP (1)
LTM CONF
Connected

Instruction elements

The following is a part of the program described above.



Logic Command

A logic command is an instruction which identifies the operation to be performed using the argument(s). In the example above, the LOAD_BIT command loads the value of the argument into an internal register called the accumulator.

There are two types of arguments:

- Setup commands
  These set up or test for the necessary conditions to perform an action (for example, LOAD and AND commands).
- Actions commands
  These commands direct the LTM R controller to perform an action based on info in the setup instructions (for example, assignment commands such as COMP).

Note: When you type a logic command, it is automatically recognized and displayed in blue.

Argument

An argument is a number, address, or bit representing a value that the LTM R controller can manipulate in an instruction. For example, in the sample program above, the second instruction "2 LOAD_BIT 683 8" includes a logic command LOAD_BIT and 2 argurments, 683 and 8. This instructs the LTM R controller to load the value of register 683 bit 8 into the accumulator. A logic command can have from zero to three arguments depending on the type of logic command.

Using instructions with commands and arguments, the LTM R controller program can:

- Define the status of controller inputs and outputs such as sensors, push buttons, and relays.
- Activate basic logic functions such as timers and counters .
- Perform arithmetic, logical, comparisons and numerical operations.
- Read or write to the LTM R controller's internal registers or to individual bits in those registers.

Note: When you type an argument, it is automatically recognized and displayed in the color assigned to the arguments.

Comments

In the Text view of the structured text editor, it is possible to add comments to the program. At the end of each line, after the arguments, typing // indicates to the program that the text that will be added is not part of the program, but is a comment.

Note: When you type //, the custom logic editor automatically recognizes the text after it as comments and displays it in green.

Synthax

You can either leave blanks between arguments or use commas or dots.

Commands

Keyboard commands and shortcuts are the same as those for Windows operating systems: press Del or Delete to delete a character or line, press Enter to go to the next line, etc.

Saving

To save the program you edited or created, click Logic Functions in the top-level menu bar and choose **Save Logic File** or **Save Logic File As**.

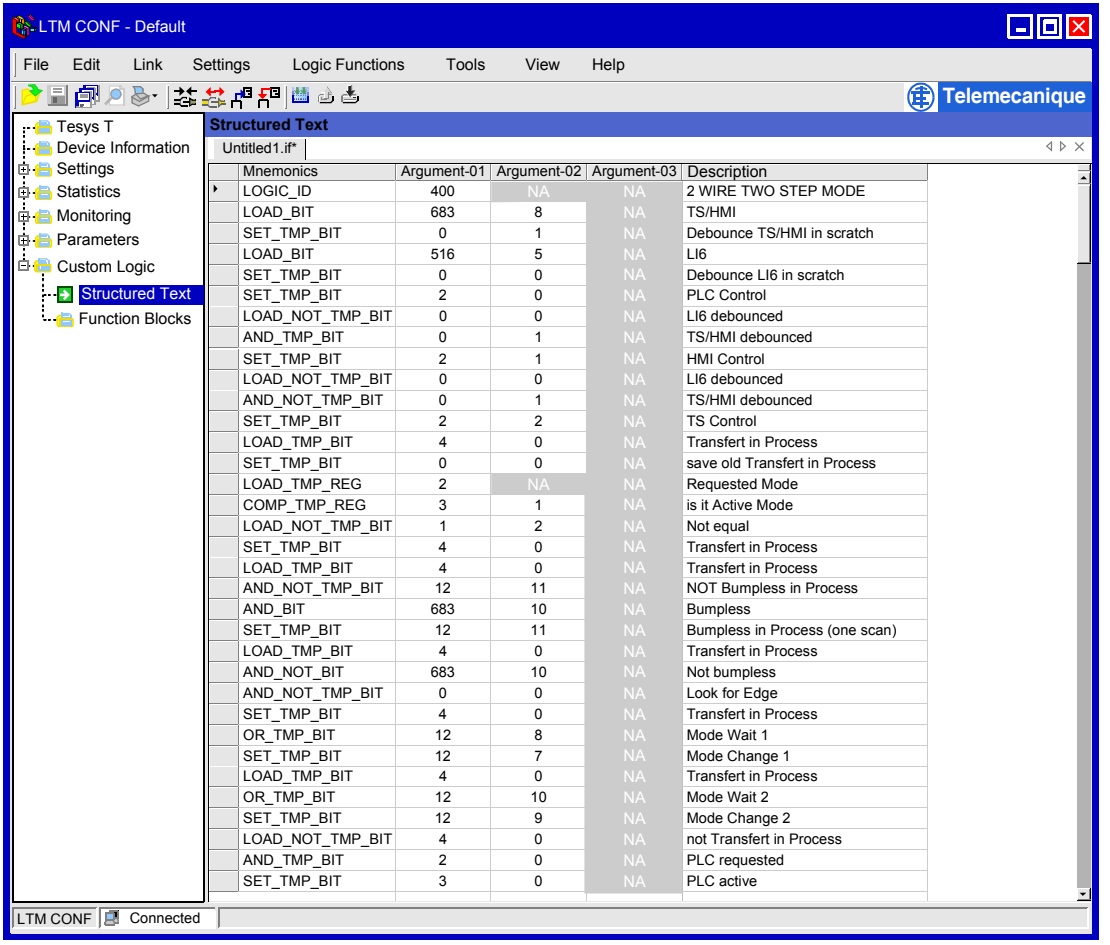
Grid View

Introduction

The structured text editor has a Grid View, which, like the text view allows you to edit or create a program. Grid view allows you to create each instruction by selecting, in a drop down menu fashion, from a list of available commands,or typing into a cell, the arguments and comments needed to complete the instruction. You may prefer this method because it is more structured using the programming rules and may help you avoid common syntax errors.

Example of a Program in Grid View

The following illustration is an example of a program in Grid view:



Mnemonics	Argument-01	Argument-02	Argument-03	Description
LOGIC_ID	400	NA	NA	2 WIRE TWO STEP MODE
LOAD_BIT	683	8	NA	TS/HMI
SET_TMP_BIT	0	1	NA	Debounce TS/HMI in scratch
LOAD_BIT	516	5	NA	LI6
SET_TMP_BIT	0	0	NA	Debounce LI6 in scratch
SET_TMP_BIT	2	0	NA	PLC Control
LOAD_NOT_TMP_BIT	0	0	NA	LI6 debounced
AND_TMP_BIT	0	1	NA	TS/HMI debounced
SET_TMP_BIT	2	1	NA	HMI Control
LOAD_NOT_TMP_BIT	0	0	NA	LI6 debounced
AND_NOT_TMP_BIT	0	1	NA	TS/HMI debounced
SET_TMP_BIT	2	2	NA	TS Control
LOAD_TMP_BIT	4	0	NA	Transfert in Process
SET_TMP_BIT	0	0	NA	save old Transfert in Process
LOAD_TMP_REG	2	NA	NA	Requested Mode
COMP_TMP_REG	3	1	NA	is it Active Mode
LOAD_NOT_TMP_BIT	1	2	NA	Not equal
SET_TMP_BIT	4	0	NA	Transfert in Process
LOAD_TMP_BIT	4	0	NA	Transfert in Process
AND_NOT_TMP_BIT	12	11	NA	NOT Bumpless in Process
AND_BIT	683	10	NA	Bumpless
SET_TMP_BIT	12	11	NA	Bumpless in Process (one scan)
LOAD_TMP_BIT	4	0	NA	Transfert in Process
AND_NOT_BIT	683	10	NA	Not bumpless
AND_NOT_TMP_BIT	0	0	NA	Look for Edge
SET_TMP_BIT	4	0	NA	Transfert in Process
OR_TMP_BIT	12	8	NA	Mode Wait 1
SET_TMP_BIT	12	7	NA	Mode Change 1
LOAD_TMP_BIT	4	0	NA	Transfert in Process
OR_TMP_BIT	12	10	NA	Mode Wait 2
SET_TMP_BIT	12	9	NA	Mode Change 2
LOAD_NOT_TMP_BIT	4	0	NA	not Transfert in Process
AND_TMP_BIT	2	0	NA	PLC requested
SET_TMP_BIT	3	0	NA	PLC active

Conversion

The software automatically converts every element in Text view to the corresponding element in Grid view.

Logic Command

The left column lists the logic commands used in the program. You can change the type of logic command by clicking a box and modifying the value in it. An arrow indicates which logic command you can choose.

Arguments

Click a value in one of the argument columns to change its value, using the up and down arrows or the number pad.

Note: If you enter a value higher than permitted, the highest permitted value will be entered.

Comments

Comments are displayed in the Description column.

Inserting an Instruction

To insert an instruction:

Step	Action
1	Choose the command box above or below the area where you want your instruction to be entered.
2	Right-click and select either "Insert row below" or "Insert row above".
3	Change the command and its assigned values to those required for your new instruction. By default, the command added in the new row is LOGIC_ID.

Deleting an Instruction

To delete an instruction:

Step	Action
1	Left-click on the command box of the row you want to delete.
2	Right-click and select "Delete row".

Logic Commands

Overview

All controller configuration files consist of a series of logic commands. Each logic command consists of the command itself, plus up to 3 arguments.

Each logic command performs its operation linked to either a 1-bit Boolean accumulator (value range 0–1) or a 16-bit unsigned accumulator (value range 0–65,535).

The custom logic editor provides the following kinds of logic commands:

- Boolean
- Register
- Timers
- Latch
- Counters
- Math

Boolean logic commands

Boolean commands evaluate and control simple Boolean (On/Off) values. Boolean commands include:

Command	Argument 1	Argument 2	Argument 3	Description
LOAD_K_BIT	Constant value (0 or 1)	-	-	Loads a constant value into the 1-bit Boolean accumulator.
LOAD_BIT	Register address	Register bit no. (0-15)	-	Loads a internal control register bit from the address identified in Argument 1, and the bit identified in Argument 2into the 1-bit Boolean accumulator.
LOAD_TMP_BIT	Temporary register address	Register bit no. (0-15)	-	Loads a temporary register bit into the 1-bit Boolean accumulator.
LOAD_NV_BIT	Non-volatile register address	Register bit no. (0-15)	-	Loads a non-volatile register bit into the 1-bit Boolean accumulator.
LOAD_NOT_BIT	Register address	Register bit no. (0-15)	-	Loads an inverted Boolean value of a register bit into the 1-bit Boolean accumulator.
LOAD_NOT_TMP_BIT	Temporary register address	Register bit no. (0-15)	-	Loads an inverted Boolean value of a temporary register bit into the 1-bit Boolean accumulator.
LOAD_NOT_NV_BIT	Non-volatile register address	Register bit no.no. (0-15)	-	Loads an inverted Boolean value of a non-volatile register bit into the 1-bit Boolean accumulator.
AND_BIT	Register address	Register bit no. (0-15)	-	Loads the result of a logical AND link between the register bit value and the accumulator content. The result is stored in the 1-bit Boolean accumulator.
AND_TMP_BIT	Temporary register address	Register bit no. (0-15)	-	Loads the result of a logical AND link between the temporary register bit value and the accumulator content. The result is stored in the 1-bit Boolean accumulator.
AND_NV_BIT	Non-volatile register address	Register bit no. (0-15)	-	Loads the result of a logical AND link between the non-volatile register bit value and the accumulator content. The result is stored in the 1-bit Boolean accumulator.
AND_NOT_BIT	Register address	Register bit no. (0-15)	-	Loads the result of a logical AND of the inverted register bit and the 1-Bit Boolean Accumulator. The result is stored in the 1-bit Boolean accumulator.
AND_NOT_TMP_BIT	Temporary register address	Register bit no. (0-15)	-	Loads the result of a logical AND of the inverted temporary register bit and the 1-Bit Boolean Accumulator. The result is stored in the 1-bit Boolean accumulator.
– Argument not applicable to logic command.				

Command	Argument 1	Argument 2	Argument 3	Description
AND_NOT_NV_BIT	Non-volatile register address	Register bit no. (0-15)	-	Loads the result of a logical AND of the inverted non-volatile register bit and the 1-Bit Boolean Accumulator. The result is stored in the 1-bit Boolean accumulator.
OR_BIT	Register address	Register bit no. (0-15)	-	Makes a logical OR link between the register bit value and the accumulator content. The result is stored in the 1-bit Boolean accumulator.
OR_TMP_BIT	Temporary register address	Register bit no. (0-15)	-	Makes a logical OR link between the temporary register bit value and the accumulator content. The result is stored in the 1-bit Boolean accumulator.
OR_NV_BIT	Non-volatile register address	Register bit no. (0-15)	-	Makes a logical OR link between the non-volatile register bit value and the accumulator content. The result is stored in the 1-bit Boolean accumulator.
OR_NOT_BIT	Register address	Register bit no.(0-15)	-	Makes an logical OR of the inverted register bit and the 1-Bit Boolean accumulator. The result is stored in the 1-bit Boolean accumulator.
OR_NOT_TMP_BIT	Temporary register address	Register bit no. (0-15)	-	Makes an logical OR of the inverted temporary register bit and the 1-Bit Boolean accumulator. The result is stored in the 1-bit Boolean accumulator.
OR_NOT_NV_BIT	Non-volatile register address	Register bit no. (0-15)	-	Makes an logical OR of the inverted non-volatile register bit and the 1-Bit Boolean accumulator. The result is stored in the 1-bit Boolean accumulator.
SET_BIT	Register address	Register bit no. (0-15)	-	Sets value of the 1-bit Boolean accumulator into a register bit.
SET_TMP_BIT	Temporary register address	Register bit no. (0-15)	-	Sets value of the 1-bit Boolean accumulator into a temporary register bit.
SET_NV_BIT	Non-volatile register address	Register bit no. (0-15)	-	Sets value of the 1-bit Boolean accumulator into a non-volatile register bit.
SET_NOT_BIT	Register address	Register bit no. (0-15)	-	Sets inverted value of the 1-bit Boolean accumulator into a register bit.
SET_NOT_TMP_BIT	Temporary register address	Register bit no. (0-15)	-	Sets inverted value of the 1-bit Boolean accumulator into a temporary register bit.
SET_NOT_NV_BIT	Non-volatile register address	Register bit no. (0-15)	-	Sets inverted value of the 1-bit Boolean accumulator into a non-volatile register bit.
– Argument not applicable to logic command.				

Register logic commands Register commands evaluate and control 16-bit values. Register commands include:

Command	Argument 1	Argument 2	Argument 3	Description
LOAD_K_REG	Constant value (0 to 65,535)	-	-	Loads a constant value into the 16-bit accumulator.
LOAD_REG	Register address	-	-	Loads a copy of a register into the 16-bit accumulator.
LOAD_TMP_REG	Temporary register address	-	-	Loads a copy of a temporary register into the 16-bit accumulator.
LOAD_NV_REG	Non-volatile register address	-	-	Loads a copy of a non-volatile register into the 16-bit accumulator.
COMP_K_REG	Constant value (0 to 65,535)	Temporary register address	-	Compares the 16-bit accumulator value to Argument 1 constant and sets status Argument 2 bits as follows: BIT 1 ON if accumulator < Argument 1 BIT 2 ON if accumulator = Argument 1 BIT 3 ON if accumulator > Argument 1
COMP_REG	Register address	Temporary register address	-	Compares the value of Argument 1 to the 16-bit accumulator content and sets status Argument 2 bits as follows: BIT 1 ON if accumulator < Argument 1 BIT 2 ON if accumulator = Argument 1 BIT 3 ON if accumulator > Argument 1
COMP_TMP_REG	Temporary register address	Temporary register address	-	Compares the value of Argument 1 to the 16-bit accumulator content and sets status Argument 2 bits as follows: BIT 1 ON if accumulator < Argument 1 BIT 2 ON if accumulator = Argument 1 BIT 3 ON if accumulator > Argument 1
COMP_NV_REG	Non-volatile register address	Temporary register address	-	Compares the 16-bit accumulator content and sets status Argument 2 bits as follows: BIT 1 ON if accumulator < Argument 1 BIT 2 ON if accumulator = Argument 1 BIT 3 ON if accumulator > Argument 1
AND_K	Constant value (0 to 65,535)	-	-	Makes a logical AND link between the constant value and the accumulator content. The result is stored in the 1-bit Boolean accumulator.
– Argument not applicable to logic command.				

Command	Argument 1	Argument 2	Argument 3	Description
AND_REG	Register address	-	-	Makes a logical AND link between the register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
AND_TMP_REG	Temporary register address	-	-	Makes a logical AND link between the temporary register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
AND_NV_REG	Non-volatile register address	-	-	Makes a logical AND link between the non-volatile register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
OR_K	Constant value (0 to 65,535)	-	-	Makes a logical OR link between the constant value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
OR_REG	Register address	-	-	Makes a logical OR link between the register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
OR_TMP_REG	Temporary register address	-	-	Makes a logical OR link between the temporary register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
OR_NV_REG	Non-volatile register address	-	-	Makes a logical exclusive OR link between the non-volatile register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
XOR_K	Constant value (0 to 65,535)	-	-	Makes a logical exclusive OR link between the constant value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
XOR_REG	Register address	-	-	Makes a logical exclusive OR link between the register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
XOR_TMP_REG	Temporary register address	-	-	Makes a logical exclusive OR link between the temporary register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
– Argument not applicable to logic command.				

Command	Argument 1	Argument 2	Argument 3	Description
AND_REG	Register address	-	-	Makes a logical AND link between the register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
AND_TMP_REG	Temporary register address	-	-	Makes a logical AND link between the temporary register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
AND_NV_REG	Non-volatile register address	-	-	Makes a logical AND link between the non-volatile register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
OR_K	Constant value (0 to 65,535)	-	-	Makes a logical OR link between the constant value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
OR_REG	Register address	-	-	Makes a logical OR link between the register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
OR_TMP_REG	Temporary register address	-	-	Makes a logical OR link between the temporary register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
OR_NV_REG	Non-volatile register address	-	-	Makes a logical exclusive OR link between the non-volatile register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
XOR_K	Constant value (0 to 65,535)	-	-	Makes a logical exclusive OR link between the constant value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
XOR_REG	Register address	-	-	Makes a logical exclusive OR link between the register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
XOR_TMP_REG	Temporary register address	-	-	Makes a logical exclusive OR link between the temporary register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
– Argument not applicable to logic command.				

Command	Argument 1	Argument 2	Argument 3	Description
XOR_NV_REG	Non-volatile register address	-	-	Makes a logical exclusive OR link between the non-volatile register value and the 16-bit accumulator content. The result is stored in the 16-bit Boolean accumulator.
ON_SET_REG	Register address	Temporary register address	-	When the 1-bit Boolean accumulator transitions from 0 to 1, the 16-bit accumulator is stored into a register (Argument 1). Status bit 3 (Argument 2) is used to remember the previous state of the 1-bit accumulator.
ON_SET_TMP_REG	Temporary register address	Temporary register address	-	When the 1-bit Boolean accumulator transitions from 0 to 1, the 16-bit accumulator is stored into a temporary register (Argument 1). Status bit 3 (Argument 2) is used to remember the previous state of the 1-bit accumulator.
ON_SET_NV_REG	Non-volatile register address	Temporary register address	-	When the 1-bit Boolean accumulator transitions from 0 to 1, the 16-bit accumulator is stored into a non-volatile register (Argument 1). Status bit 3 (Argument 2) is used to remember the previous state of the 1-bit accumulator.
– Argument not applicable to logic command.				

Timer logic commands

Timers have a range of 0 to 65 535 and measure time in intervals of seconds or tenths of seconds. Argument 1 specifies the time period; Argument 2 is a calculated end time. The first four bits of the status register (Argument 3) describe timer operations, as follows:

- bit 0: enable input bit; the rising edge of this bit starts the timer and sets bit 2
- bit 1: timed out status bit; cleared by clearing bit 0 or by cycling power
- bit 2: timing status bit; expiration of the time period clears bit 2 and sets bit 1
- bit 3: enable history.

Timer commands include:

Command	Argument 1	Argument 2	Argument 3	Description
TIMER_SEC	Temporary register (time period)	Temporary register (calculated end time)	Temporary register (status)	Counts in seconds the time period input in Arg1 as described by status register bits: 0=Enable; 1=Timed Out; 2=Timing; 3=Enable History.
TIMER_TENTHS	Temporary register (time period)	Temporary register (calculated end time)	Temporary register (status)	Counts in tenths of seconds the time period input in Arg1 as described by status register bits: 0=Enable; 1=Timed Out; 2=Timing; 3=Enable History.
TIMER_K_SEC	Constant value 0 to 65,535 (time period)	Temporary register (calculated end time)	Temporary register (status)	Counts in seconds the time period input in Arg1 as described by status register bits: 0=Enable; 1=Timed Out; 2=Timing; 3=Enable History.
TIMER_K_TENTHS	Constant value 0 to 65535 (time period)	Temporary register (calculated end time)	Temporary register (status)	Counts in tenths of seconds the time period input in Arg1 as described by status register bits: 0=Enable; 1=Timed Out; 2=Timing; 3=Enable History.

Latch logic commands

Latch commands include:

Command	Argument 1	Argument 2	Argument 3	Description
LATCH	Temporary register (status)	-	-	Records and retains in a temporary register a history of a signal. Status bits: 0=State; 1=Set; 2=Clear; 3=Set History; 4=Clear History.
LATCH_NV	Non-volatile register (status)	-	-	Records and retains in a non-volatile register a history of a signal. Status bits: 0=State; 1=Set; 2=Clear; 3=Set History; 4=Clear History.
– Argument not applicable to logic command.				

Counter logic commands

Counters have a range of 0 to 65 535 and transition to 0 upon counting to the maximum value of 65 535. Counters perform a comparison between the counted value (Arg1) and a constant value (Arg2). The first 10 bits of the status register (Arg3) describe counter operations, as follows:

- bit 0: indicates that the count (Arg1) is 0
- bit 1: indicates that the count (Arg1) is less than the constant value (Arg2)
- bit 2: indicates that the count (Arg1) is equal to the constant value (Arg2)
- bit 3: indicates that the count (Arg1) is greater than the constant value (Arg2)
- bit 4: increments counter (Arg1) on detection of rising edge of input
- bit 5: decrements counter (Arg1) on detection of rising edge of input
- bit 6: sets counter (Arg1) equal to constant value (Arg2)
- bit 7: increment history
- bit 8: decrement history
- bit 9: set history

Latch commands include:

Command	Argument 1	Argument 2	Argument 3	Description
COUNTER	Temporary register (count)	Constant value 0 to 65,535 (initial value)	Temporary register (status)	Performs a comparative count, saving both the count and status to temporary registers.
COUNTER_NV	Non-volatile register (count)	Constant value 0 to 65,535 (initial value)	Non-volatile register (status)	Performs a comparative count, saving both the count and status to non-volatile registers.

Math logic commands

Math commands perform unsigned math functions using the 16-bit accumulator and temporary registers. Math commands are executed when the 1-bit accumulator transitions from 0 to 1. Math commands include:

Command	Argument 1	Argument 2	Argument 3	Description
ON_ADD	Temporary register (value)	Temporary register (status)	-	Arg1=Arg1+16-bit accumulator. Status bits: 0=Overflow; 3=On History
ON_SUB	Temporary register (value)	Temporary register (status)	-	Arg1=Arg1-16-bit accumulator. Status bits: 0=Underflow; 3=On History
ON_MUL	Temporary register (most significant word)	Temporary register (least significant word)	Temporary register (status)	Arg1:Arg2=16-bit accumulator x Arg2. Status bits: 3=On History
ON_DIV	Temporary register (most significant word)	Temporary register (least significant word)	Temporary register (status)	Arg1:Arg2=Arg1:Arg2 / 16-bit accumulator. Status bits: 3=On History
– Argument not applicable to logic command.				

2.2 Logic Commands

Logic Commands Overview

Summary

This section describes in detail the logic commands and arguments provided by the custom logic editor.

What's in this Section?

This section contains the following topics:

Topic	Page
Program Logic Commands	54
Boolean Logic Commands	55
Register Logic Commands	69
Timer Logic Commands	85
Latch Logic Commands	89
Counter Logic Commands	91
Math Logic Commands	93

Program Logic Commands

Overview

Program logic commands are used to identify the logic file to the custom logic editor.

The following commands can be used:

- LOGIC_ID
- NOP

LOGIC_ID

The LOGIC_ID statement acts as an identifier for the logic file.

LOGIC_ID (see *p. 17*) values have an integer value range of 0 to 511, as follows:

- 0 - 255: reserved for default logic files
- 256 - 511: available for custom logic files.

Arguments	Representation
1	`LOGIC_ID ID#`

Input argument:		
Argument	**Type**	**Description**
ID#	UINT	An integer from 0 to 511.
Output arguments:		
-		

NOP

The NOP command performs no operation.

Use the NOP command as a placeholder in a logic file to replace a pre-existing command, or to reserve space for a future command.

Arguments	Representation
0	`NOP`

The NOP command has no arguments.

Boolean Logic Commands

Overview

The custom logic editor uses the following boolean logic commands:

- LOAD_K_BIT
- LOAD_BIT
- LOAD_TMP_BIT
- LOAD_NV_BIT
- LOAD_NOT_BIT
- LOAD_NOT_TMP_BIT
- LOAD_NOT_NV_BIT
- AND_BIT
- AND_TMP_BIT
- AND_NV_BIT
- AND_NOT_BIT
- AND_NOT_TMP_BIT
- AND_NOT_NV_BIT
- OR_BIT
- OR_TMP_BIT
- OR_NV_BIT
- OR_NOT_BIT
- OR_NOT_TMP_BIT
- OR_NOT_NV_BIT
- SET_BIT
- SET_TMP_BIT
- SET_NV_BIT
- SET_NOT_BIT
- SET_NOT_TMP_BIT
- SET_NOT_NV_BIT

LOAD_K_BIT

The LOAD_K_BIT command loads a constant Boolean value (0 or 1) into the 1-bit Boolean accumulator.

Arguments	Representation
1	`LOAD_K_BIT KValue`

Input arguments:		
Argument	**Type**	**Description**
KValue	BOOL	A constant value 0 or 1.
Output arguments:		
-		

LOAD_BIT

The LOAD_BIT command loads the Boolean value (0 or 1) of a register bit into the the 1-bit Boolean accumulator.

Arguments	Representation
2	`LOAD_BIT RegAddr BitNo`

Input arguments		
Argument	**Type**	**Description**
RegAddr	UINT	The register address: an integer from 0 to 1399.
BitNo	UINT	The bit number: an integer from 0 to 15.
Output arguments:		
-		

LOAD_TMP_BIT The LOAD_TMP_BIT command loads the Boolean value (0 or 1) of a temporary register bit into the 1-bit Boolean accumulator.

Arguments	Representation
2	`LOAD_TMP_BIT TmpReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	The temporary register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic Temporary Space register at address 1204.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

LOAD_NV_BIT The LOAD_NV_BIT command loads the Boolean value (0 or 1) of a non-volatile register bit into the 1-bit Boolean accumulator.

Arguments	Representation
2	`LOAD_NV_BIT NVReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
NVReg	UINT	The non-volatile space register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic NonVolatile Space register at address 1205.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

LOAD_NOT_BIT The LOAD_NOT_BIT command:

- inverts the Boolean value (0 or 1) of a specified register bit, then
- loads that value into the 1-bit Boolean accumulator.

Arguments	Representation
2	`LOAD_NOT_BIT RegAddr BitNo`

Input arguments:		
Argument	**Type**	**Description**
RegAddr	UINT	The register address: an integer from 0 to 1399.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

LOAD_NOT_TMP_BIT The LOAD_NOT_TMP_BIT command:

- inverts the Boolean value (0 or 1) of a specified temporary register bit, then
- loads it into the 1-bit Boolean accumulator.

Arguments	Representation
2	`LOAD_NOT_TMP_BIT TmpReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	The temporary register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic Temporary Space register at address 1204.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

LOAD_NOT_NV_BIT The LOAD_NOT_NV_BIT command:

- inverts the Boolean value (0 or 1) of a selected non-volatile register bit, then
- loads it into the 1-bit Boolean accumulator.

Arguments	Representation
2	`LOAD_NOT_NV_BIT NVReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
NVReg	UINT	The non-volatile space register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic NonVolatile Space register at address 1205.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

AND_BIT The AND_BIT command makes a logical AND link between a register bit value and the accumulator content in logic memory.

If the bit accumulator equals 1 and the linked register bit equals 1, the result of the AND process is also 1; in all other cases the result of the AND process is 0. The result is saved in the 1-bit accumulator.

Arguments	Representation
2	`AND_BIT RegAddr BitNo`

Input arguments		
Argument	**Type**	**Description**
RegAddr	UINT	The register address: an integer from 0 to 1399.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

AND_TMP_BIT

The AND_TMP_BIT command makes a logical AND link between a temporary register bit value and the accumulator content in logic memory.

If the bit accumulator equals 1 and the linked temporary register bit equals 1, the result of the AND process is also 1; in all other cases the result of the AND process is 0. The result is saved in the 1-bit accumulator.

Arguments	Representation
2	`AND_TMP_BIT TmpReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	The temporary register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic Temporary Space register at address 1204.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

AND_NV_BIT

The AND_NV_BIT command makes a logical AND link between a non-volatile register bit value and the accumulator content in logic memory.

If the bit accumulator equals 1 and the linked non-volatile register bit equals 1, the result of the AND process is also 1; in all other cases the result of the AND process is 0. The result is saved in the 1-bit accumulator.

Arguments	Representation
2	`AND_NV_BIT NVReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
NVReg	UINT	The non-volatile space register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic NonVolatile Space register at address 1205.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

AND_NOT_BIT

The AND_NOT_BIT command:

- inverts the Boolean value (0 or 1) of a specified register bit, then
- makes a logical AND link between it and the accumulator content in logic memory.

If the bit accumulator equals 1 and the inverted linked register bit equals 1, the result of the AND process is also 1; in all other cases the result of the AND process is 0. The result is saved in the 1-bit accumulator.

Arguments	Representation
2	`AND_NOT_BIT RegAddr BitNo`

Input arguments:		
Argument	**Type**	**Description**
RegAddr	UINT	The register address: an integer from 0 to 1399.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

AND_NOT_TMP_BIT

The AND_NOT_TMP_BIT command:

- inverts the Boolean value (0 or 1) of a specified temporary register bit, then
- makes a logical AND link between it and the accumulator content in logic memory.

If the bit accumulator equals 1 and the inverted linked temporary register bit equals 1, the result of the AND process is also 1; in all other cases the result of the AND process is 0. The result is saved in the 1-bit accumulator.

Arguments	Representation
2	`AND_NOT_TMP_BIT TmpReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	The temporary register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic Temporary Space register at address 1204.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

AND_NOT_NV_BIT

The AND_NOT_NV_BIT command:

- inverts the Boolean value (0 or 1) of a selected non-volatile register bit, then
- makes a logical AND link between it and the accumulator content in logic memory.

If the bit accumulator equals 1 and the linked non-volatile register bit equals 1, the result of the AND process is also 1; in all other cases the result of the AND process is 0. The result is saved in the 1-bit accumulator.

Arguments	Representation
2	`AND_NOT_NV_BIT NVReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
NVReg	UINT	The non-volatile space register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic NonVolatile Space register at address 1205.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

OR_BIT

The OR_BIT command makes a logical OR link between a register bit value and the accumulator content in logic memory.

If the value of either the bit accumulator or the register bit equals 1, the result of the OR process is also 1; if the values of all compared bits equal 0, the result of the OR process is 0. The result is saved in the 1-bit accumulator.

Arguments	Representation
2	`OR_BIT RegAddr BitNo`

Input arguments		
Argument	**Type**	**Description**
RegAddr	UINT	The register address: an integer from 0 to 1399.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

OR_TMP_BIT

The OR_TMP_BIT command makes a logical OR link between a temporary register bit value and the accumulator content in logic memory.

If the value of either the bit accumulator or the temporary register bit equals 1, the result of the OR process is also 1;if the values of all compared bits equal 0, the result of the OR process is 0. The result is saved in the 1-bit accumulator.

Arguments	Representation
2	`OR_TMP_BIT TmpReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	The temporary register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic Temporary Space register at address 1204.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

OR_NV_BIT

The OR_NV_BIT command makes a logical OR link between a non-volatile register bit value and the accumulator content in logic memory.

If the value of either the bit accumulator or the non-volatile register bit equals 1, the result of the OR process is also 1;if the values of all compared bits equal 0, the result of the OR process is 0. The result is saved in the 1-bit accumulator.

Arguments	Representation
2	`OR_NV_BIT NVReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
NVReg	UINT	The non-volatile space register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic NonVolatile Space register at address 1205.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

OR_NOT_BIT

The OR_NOT_BIT command:

- inverts the Boolean value (0 or 1) of a specified register bit, then
- makes a logical OR link between it and the accumulator content in logic memory.

If the value of either the bit accumulator or the inverted register bit equals 1, the result of the OR process is also 1; if the values of all compared bits equal 0, the result of the OR process is 0. The result is saved in the 1-bit accumulator.

Arguments	Representation
2	`OR_NOT_BIT RegAddr BitNo`

Input arguments:		
Argument	**Type**	**Description**
RegAddr	UINT	The register address: an integer from 0 to 1399.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

OR_NOT_TMP_BIT The OR_NOT_TMP_BIT command:

- inverts the Boolean value (0 or 1) of a specified temporary register bit, then
- makes a logical OR link between it and the accumulator content in logic memory.

If the value of either the bit accumulator or the inverted temporary register bit equals 1, the result of the OR process is also 1; if the values of all compared bits equal 0, the result of the OR process is 0. The result is saved in the 1-bit accumulator.

Arguments	Representation
2	`OR_NOT_TMP_BIT TmpReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	The temporary register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic Temporary Space register at address 1204.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

OR_NOT_NV_BIT The OR_NOT_NV_BIT command:

- inverts the Boolean value (0 or 1) of a selected non-volatile register bit, then
- makes a logical OR link between it and the accumulator content in logic memory.

If the value of either the bit accumulator or the inverted non-volatile register bit equals 1, the result of the OR process is also 1; if the values of all compared bits equal 0, the result of the OR process is 0. The result is saved in the 1-bit accumulator.

Arguments	Representation
2	`OR_NOT_NV_BIT NVReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
NVReg	UINT	The non-volatile space register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic NonVolatile Space register at address 1205.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

SET_BIT

The SET_BIT command sets the value of the 1-bit accumulator to a specified register bit.

Arguments	Representation
2	`SET_BIT RegAddr BitNo`

Input arguments		
Argument	**Type**	**Description**
RegAddr	UINT	The register address: an integer from 0 to 1399.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

SET_TMP_BIT

The SET_TMP_BIT command sets the value of the 1-bit accumulator to a specified temporary register bit.

Arguments	Representation
2	`SET_TMP_BIT TmpReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	The temporary register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic Temporary Space register at address 1204.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

SET_NV_BIT

The SET_NV_BIT command sets the value of the 1-bit accumulator to a specified non-volatile register bit.

Arguments	Representation
2	`SET_NV_BIT NVReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
NVReg	UINT	The non-volatile space register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic NonVolatile Space register at address 1205.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

SET_NOT_BIT

The SET_NOT_BIT command sets the inverted value of the 1-bit accumulator to a specified register bit.

Arguments	Representation
2	`SET_NOT_BIT RegAddr BitNo`

Input arguments:		
Argument	**Type**	**Description**
RegAddr	UINT	The register address: an integer from 0 to 1399.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

SET_NOT_TMP_BIT The SET_NOT_TMP_BIT command sets the inverted value of the 1-bit accumulator to a specified temporary register bit.

Arguments	Representation
2	`SET_NOT_TMP_BIT TmpReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	The temporary register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic Temporary Space register at address 1204.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

SET_NOT_NV_BIT The SET_NOT_NV_BIT command sets the inverted value of the 1-bit accumulator to a specified non-volatile register bit.

Arguments	Representation
2	`SET_NOT_NV_BIT NVReg BitNo`

Input arguments:		
Argument	**Type**	**Description**
NVReg	UINT	The non-volatile space register number. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic NonVolatile Space register at address 1205.
BitNo	UINT	The bit location: an integer from 0 to 15.
Output arguments:		
-		

Register Logic Commands

Overview

Register commands evaluate and control 16-bit values.

The custom logic editor uses the following register commands:

- LOAD_K_REG
- LOAD_REG
- LOAD_TMP_REG
- LOAD_NV_REG
- COMP_K_REG
- COMP_REG
- COMP_TMP_REG
- COMP_NV_REG
- AND_K
- AND_REG
- AND_TMP_REG
- AND_NV_REG
- OR_K
- OR_REG
- OR_TMP_REG
- OR_NV_REG
- XOR_K
- XOR_REG
- XOR_TMP_REG
- XOR_NV_REG
- ON_SET_REG
- ON_SET_TMP_REG
- ON_SET_NV_REG

LOAD_K_REG

The LOAD_K_REG command loads a constant 16-bit value into the accumulator in logic memory.

Arguments	Representation
1	`LOAD_K_REG KValue`

Input arguments:		
Argument	**Type**	**Description**
KValue	UINT	A constant value from 0 to 65,535.
Output arguments:		
-		

LOAD_REG

The LOAD_REG command loads a copy of a register into the accumulator in logic memory.

Arguments	Representation
1	`LOAD_REG RegAddr`

Input arguments:		
Argument	**Type**	**Description**
RegAddr	UINT	The register address: an integer from 0 to 1399.
Output arguments:		
-		

LOAD_TMP_REG The LOAD_TMP_REG command loads a copy of a temporary register into the accumulator in logic memory.

Arguments	Representation
1	`LOAD_TMP_REG TmpReg`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	The temporary register number: An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic Temporary Space register at address 1204.
Output arguments:		
-		

LOAD_NV_REG The LOAD_NV_REG command loads a copy of a non-volatile register into the accumulator in logic memory.

Arguments	Representation
1	`LOAD_NV_REG NVReg`

Input arguments:		
Argument	**Type**	**Description**
NVReg	UINT	The non-volatile space register number: an integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic NonVolatile Space register at address 1205.
Output arguments:		
-		

COMP_K_REG

The COMP_K_REG command compares the accumulator content to the Argument 1 constant value and sets one of the following bits in a temporary register:

- bit 0 (not used)
- bit 1 if the accumulator is less than the constant value
- bit 2 if the accumulator equals the constant value
- bit 3 if the accumulator is greater than the constant value.

Arguments	Representation
2	`COMP_K_REG KValue TmpReg`

Input arguments:		
Argument	**Type**	**Description**
KValue	UINT	Argument 1: a constant value from 0 to 65,535.
Output arguments:		
LT	BOOL	Argument 2.Bit 1: accumulator < constant.
EQ	BOOL	Argument 2.Bit 2: accumulator = constant.
GT	BOOL	Argument 2.Bit 3: accumulator > constant.

COMP_REG

The COMP_REG command compares the accumulator content to the value of the Argument 1 register and sets one of the following bits in a temporary register:

- bit 0 (not used)
- bit 1 if the accumulator is less than the register value
- bit 2 if the accumulator equals the register value
- bit 3 if the accumulator is greater than the register value.

Arguments	Representation
2	`COMP_REG RegAddr TmpReg`

Input arguments:		
Argument	**Type**	**Description**
RegAddr	UINT	Argument 1: any valid LTM R register.
Output arguments:		
LT	BOOL	Argument 2.Bit 1: accumulator < register.
EQ	BOOL	Argument 2.Bit 2: accumulator = register.
GT	BOOL	Argument 2.Bit 3: accumulator > register.

COMP_TMP_REG

The COMP_TMP_REG command compares the accumulator content to the value of the Argument 1 temporary register and sets one of the following bits in a temporary register:

- bit 0 (not used)
- bit 1 if the accumulator is less than the temporary register value
- bit 2 if the accumulator equals the temporary register value
- bit 3 if the accumulator is greater than the temporary register value.

Arguments	Representation
2	`COMP_TMP_REG TmpReg TmpReg`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	Argument 1 temporary register number: An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic Temporary Space register at address 1204.
Output arguments:		
LT	BOOL	Argument 2.Bit 1: accumulator < temporary register.
EQ	BOOL	Argument 2.Bit 2: accumulator = temporary register.
GT	BOOL	Argument 2.Bit 3: accumulator > temporary register.

COMP_NV_REG

The COMP_NV_REG command compares the accumulator content to the value of the Argument 1 non-volatile register and sets one of the following bits in a temporary register:

- bit 0 (not used)
- bit 1 if the accumulator is less than the non-volatile register value
- bit 2 if the accumulator equals the non-volatile register value
- bit 3 if the accumulator is greater than the non-volatile register value.

Arguments	Representation
2	`COMP_NV_REG NVReg TmpReg`

Input arguments:		
Argument	**Type**	**Description**
NVReg	UINT	Argument 1 non-volatile register number: an integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic NonVolatile Space register at address 1205.
Output arguments:		
NA	BOOL	Argument 2.Bit 0: comparison of the register to the accumulator is invalid.
LT	BOOL	Argument 2.Bit 1: register < accumulator.
EQ	BOOL	Argument 2.Bit 2: register = accumulator.
GT	BOOL	Argument 2.Bit 3: register > accumulator.

AND_K

The AND_K command makes a logical AND link between a 16-bit constant value and the accumulator content in logic memory.

The AND process compares each bit in the 16-bit accumulator with the corresponding bit in the linked 16-bit constant. If both bits equal 1, the result of the AND process for that bit location is also 1; in all other cases the result of the AND process for that bit location is 0. The result is saved in the 16-bit accumulator.

Arguments	Representation
1	`AND_K KValue`

Input arguments:		
Argument	**Type**	**Description**
KValue	UINT	A constant value from 0 to 65,535.
Output arguments:		
-		

AND_REG

The AND_REG command makes a logical AND link between the register value and the accumulator content in logic memory.

The AND process compares each bit in the 16-bit accumulator with the corresponding bit in the linked register. If both bits equal 1, the result of the AND process for that bit location is also 1; in all other cases the result of the AND process for that bit location is 0. The result is saved in the 16-bit accumulator.

Arguments	Representation
1	`AND_REG RegAddr`

Input arguments:		
Argument	**Type**	**Description**
RegAddr	UINT	The register address: any valid LTM R register.
Output arguments:		
-		

AND_TMP_REG

The AND_TMP_REG command makes a logical AND link between the temporary register value and the accumulator content in logic memory.

The AND process compares each bit in the 16-bit accumulator with the corresponding bit in the linked temporary register. If both bits equal 1, the result of the AND process for that bit location is also 1; in all other cases the result of the AND process for that bit location is 0. The result is saved in the 16-bit accumulator.

Arguments	Representation
1	`AND_TMP_REG TmpReg`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	The temporary register number: an integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic Temporary Space register at address 1204.
Output arguments:		
-		

AND_NV_REG

The AND_NV_REG command makes a logical AND link between the non-volatile register value and the accumulator content in logic memory.

The AND process compares each bit in the 16-bit accumulator with the corresponding bit in the linked non-volatile register. If both bits equal 1, the result of the AND process for that bit location is also 1; in all other cases the result of the AND process for that bit location is 0. The result is saved in the 16-bit accumulator.

Arguments	Representation
1	`AND_NV_REG NVReg`

Input arguments:		
Argument	**Type**	**Description**
NVReg	UINT	The non-volatile space register number: an integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic NonVolatile Space register at address 1205.
Output arguments:		
-		

OR_K

The OR_K command makes a logical OR link between a 16-bit constant value and the accumulator content in logic memory.

The OR process compares each bit in the 16-bit accumulator with the corresponding bit in the linked 16-bit constant. If any compared bit equals 1, the result of the OR process for that bit location is also 1; if all compared bits equal 0, the result of the OR process for that bit location is 0. The result is saved in the 16-bit accumulator.

Arguments	Representation
1	`OR_K KValue`

Input arguments:		
Argument	**Type**	**Description**
KValue	UINT	A constant value from 0 to 65,535.
Output arguments:		
-		

OR_REG

The OR_REG command makes a logical OR link between the register value and the accumulator content in logic memory.

The OR process compares each bit in the 16-bit accumulator with the corresponding bit in the linked register. If any compared bit equals 1, the result of the OR process for that bit location is also 1; if all compared bits equal 0, the result of the OR process for that bit location is 0. The result is saved in the 16-bit accumulator.

Arguments	Representation
1	`OR_REG RegAddr`

Input arguments:		
Argument	**Type**	**Description**
RegAddr	UINT	The register address: any valid LTM R register.
Output arguments:		
-		

OR_TMP_REG

The OR_TMP_REG command makes a logical OR link between the temporary register value and the accumulator content in logic memory.

The OR process compares each bit in the 16-bit accumulator with the corresponding bit in the linked temporary register. If any compared bit equals 1, the result of the OR process for that bit location is also 1; if all compared bits equal 0, the result of the OR process for that bit location is 0. The result is saved in the 16-bit accumulator.

Arguments	Representation
1	`OR_TMP_REG TmpReg`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	The temporary register number: an integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic Temporary Space register at address 1204.
Output arguments:		
-		

OR_NV_REG

The OR_NV_REG command makes a logical OR link between the non-volatile register value and the accumulator content in logic memory.

The OR process compares each bit in the 16-bit accumulator with the corresponding bit in the linked non-volatile register. If any compared bit equals 1, the result of the OR process for that bit location is also 1; if all compared bits equal 0, the result of the OR process for that bit location is 0. The result is saved in the 16-bit accumulator.

Arguments	Representation
1	`OR_NV_REG NVReg`

Input arguments:		
Argument	**Type**	**Description**
NVReg	UINT	The non-volatile space register number: an integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic NonVolatile Space register at address 1205.
Output arguments:		
-		

XOR_K

The XOR_K command makes a logical OR link between a 16-bit constant value and the accumulator content in logic memory. The result is saved in the 16-bit accumulator.

The XOR process compares each bit in the 16-bit accumulator with the corresponding bit in the linked 16-bit constant and yields these results:

When 2 bits are compared:	
if one bit equals 1 and the other equals 0...	the result of the XOR process is 1.
in all other cases...	the result of the XOR process is 0.
When more than 2 bits are compared:	
if there is an odd number of 1 states...	the result of the XOR process is 1.
if there is an even number of 1 states...	the result of the XOR process is 0.

Arguments	**Representation**
1	`OXR_K KValue`

Input arguments:		
Argument	**Type**	**Description**
KValue	UINT	A constant value from 0 to 65,535.
Output arguments:		
-		

XOR_REG

The XOR_REG command makes a logical exclusive OR link between the register value and the accumulator content in logic memory. The result is saved in the 16-bit accumulator.

The XOR process compares each bit in the 16-bit accumulator with the corresponding bit in the linked register and yields these results:

When 2 bits are compared:	
if one bit equals 1 and the other bit equals 0...	the result of the XOR process is 1.
in all other cases...	the result of the XOR process is 0.
When more than 2 bits are compared:	
if there is an odd number of 1 states...	the result of the XOR process is 1.
if there is an even number of 1 states...	the result of the XOR process is 0.

Arguments	**Representation**
1	`XOR_REG RegAddr`

Input arguments:		
Argument	**Type**	**Description**
RegAddr	UINT	The register address: any valid LTM R register.
Output arguments:		
-		

XOR_TMP_REG

The XOR_TMP_REG command makes a logical exclusive OR link between the temporary register value and the accumulator content in logic memory. The result is saved in the 16-bit accumulator.

The XOR process compares each bit in the 16-bit accumulator with the corresponding bit in the linked temporary register and yields these results:

When 2 bits are compared:	
if one bit equals 1 and the other equals 0...	the result of the XOR process is 1.
in all other cases...	the result of the XOR process is 0.
When more than 2 bits are compared:	
if there is an odd number of 1 states...	the result of the XOR process is 1.
if there is an even number of 1 states...	the result of the XOR process is 0.

Arguments	**Representation**
1	`XOR_TMP_REG TmpReg`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	The temporary register number: an integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic Temporary Space register at address 1204.
Output arguments:		
-		

XOR_NV_REG

The XOR_NV_REG command makes a logical XOR link between the non-volatile register value and the accumulator content in logic memory. The result is saved in the 16-bit accumulator.

The XOR process compares each bit in the 16-bit accumulator with the corresponding bit in the linked non-volatile register and yields these results:

When 2 bits are compared:	
if one bit equals 1 and the other equals 0...	the result of the XOR process is 1.
in all other cases...	the result of the XOR process is 0.
When more than 2 bits are compared:	
if there is an odd number of 1 states...	the result of the XOR process is 1.
if there is an even number of 1 states...	the result of the XOR process is 0.

Arguments	**Representation**
1	`XOR_NV_REG NVReg`

Input arguments:		
Argument	**Type**	**Description**
NVReg	UINT	The non-volatile space register number: an integer value ranging from 0 to 1 less than the value of the Custom Logic NonVolatile Space register at address 1205.
Output arguments:		
-		

ON_SET_REG

The ON_SET_REG command copies the value of the 16-bit accumulator to a register on detecting the rising edge of an input signal that sets the bit accumulator value to 1 when the OnHistory bit value is 0.

The OnHistory bit holds the value of the bit accumulator (0 or 1) from the previous scan.

Arguments	Representation
2	`ON_SET_REG RegAddr TmpReg`

Input arguments:		
Argument	**Type**	**Description**
-		
Output arguments:		
RegAddr	UINT	The address of the register to be set: any valid writable LTM R register.
OnHistory	BOOL	Argument 2.Bit 3: contains the bit accumulator value from the previous scan.

ON_SET_TMP_REG

The ON_SET_TMP_REG command copies the value of the 16-bit accumulator to a temporary register to 1 on detecting the rising edge of an input signal that sets the bit accumulator value when the OnHistory bit value is 0.

The OnHistory bit holds the value of the bit accumulator (0 or 1) from the previous scan.

Arguments	Representation
2	`ON_SET_TMP_REG TmpReg TmpReg`

Input arguments:		
Argument	**Type**	**Description**
-		
Output arguments:		
RegAddr	UINT	Argument 1: the address of the register to be set. An integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic Temporary Space register at address 1204.
OnHistory	BOOL	Argument 2.Bit 3: contains the bit accumulator value from the previous scan.

ON_SET_NV_REG

The ON_SET_NV_REG command copies the value of the 16-bit accumulator to a non-volatile register to 1 on detecting the rising edge of an input signal that sets the bit accumulator value to 1 when the OnHistory bit value is 0.

The OnHistory bit holds the value of the bit accumulator (0 or 1) from the previous scan.

Arguments	Representation
1	`ON_SET_NV_REG NVReg NVReg`

Input arguments:		
Argument	**Type**	**Description**
-		
Output arguments:		
RegAddr	UINT	Argument 1: an integer ranging from 0 to the value equalling 1 less than the value of the Custom Logic NonVolatile Space register at address 1205.
OnHistory	BOOL	Argument 2.Bit 3: contains the bit accumulator value from the previous scan.

Timer Logic Commands

Overview

The custom logic editor uses the following Timer commands:

- TIMER_SEC
- TIMER_MS
- TIMER_K_SEC
- TIMER_K_MS

TIMER_SEC

The TIMER_SEC command:

- counts time in seconds, up to the number of counts specified by a temporary register
- calculates and tracks the time remaining in a 2nd temporary register
- is enabled by, and reports its counting status to, a 3rd temporary register.

Arguments	Representation
3	`TIMER_SEC TmpReg TmpReg TmpReg`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	Argument 1: the number of counts. An integer from 0 to 65,535.
Enable	BOOL	Argument 3.Bit 0: the rising edge of this bit starts the timer.
Output arguments:		
EndTime	UINT	Argument 2: a calculation of the time remaining. An integer from 0 to 65,535.
TimedOut	BOOL	Argument 3.Bit 1: indicates that timing has stopped. This bit is set when Argument 2 expires. This bit is cleared when: • Argument 3. Bit 0 is cleared • power is cycled.
Timing	BOOL	Argument 3.Bit 2: indicates that timing is ongoing. This bit is cleared when Argument 2 expires.
EnableHistory	BOOL	Argument 3.Bit 3: Status of the Enable bit in the previous scan.

TIMER_TENTHS

The TIMER_TENTHS command:

- counts time in tenths of seconds, up to the number of counts specified by a temporary register
- calculates and tracks the time remaining in a 2nd temporary register
- is enabled by, and reports its counting status to, a 3rd temporary register.

Arguments	Representation
3	`TIMER_MS TmpReg TmpReg TmpReg`

Input arguments:		
Argument	**Type**	**Description**
TmpReg	UINT	Argument 1: the number of counts. An integer from 0 to 65,535.
Enable	BOOL	Argument 3.Bit 0: the rising edge of this bit starts the timer.
Output arguments:		
EndTime	UINT	Argument 2: a calculation of the time remaining. An integer from 0 to 65,535.
TimedOut	BOOL	Argument 3.Bit 1: indicates that timing has stopped. This bit is set when Argument 2 expires. This bit is cleared when: • Argument 3.Bit 0 is cleared • power is cycled.
Timing	BOOL	Argument 3.Bit 2: indicates that timing is ongoing. This bit is cleared when Argument 2 expires.
EnableHistory	BOOL	Argument 3.Bit 3: status of the Enable bit in the previous scan.

TIMER_K_SEC The TIMER_K_SEC command:

- counts time in seconds, up to the number of counts specified by a constant value
- calculates and tracks the time remaining in a temporary register
- is enabled by, and reports its counting status to, a 2nd temporary register.

Arguments	Representation
3	`TIMER_K_SEC KValue TmpReg TmpReg`

Input arguments:		
Argument	**Type**	**Description**
KValue	UINT	Argument 1: the number of counts. An integer value from 0 to 65,535.
Enable	BOOL	Argument 3.Bit 0: the rising edge of this bit starts the timer.
Output arguments:		
EndTime	UINT	Argument 2: a calculation of the time remaining. An integer from 0 to 65,535.
TimedOut	BOOL	Argument 3.Bit 1: indicates that timing has stopped. This bit is set when Argument 2 expires. This bit is cleared when: • Argument 3. Bit 0 is cleared • power is cycled.
Timing	BOOL	Argument 3.Bit 2: indicates that timing is ongoing. This bit is cleared when Argument 2 expires.
EnableHistory	BOOL	Argument 3.Bit 3: Status of the Enable bit in the previous scan.

TIMER_K_TENTHS The TIMER_K_TENTHS command:

- counts time in tenths of seconds, up to the number of counts specified by a constant value
- calculates and tracks the time remaining in a temporary register
- is enabled by, and reports its counting status to, a 2nd temporary register.

Arguments	Representation
3	`TIMER_K_MS KValue TmpReg TmpReg`

Input arguments:		
Argument	**Type**	**Description**
KValue	UINT	Argument 1: the number of counts. An integer from 0 to 65,535.
Enable	BOOL	Argument 3.Bit 0: the rising edge of this bit starts the timer.
Output arguments:		
EndTime	UINT	Argument 2: a calculation of the time remaining. An integer from 0 to 65,535.
TimedOut	BOOL	Argument 3.Bit 1: indicates that timing has stopped. This bit is set when Argument 2 expires. This bit is cleared when: • Argument 3. Bit 0 is cleared • power is cycled.
Timing	BOOL	Argument 3.Bit 2: indicates that timing is ongoing. This bit is cleared when Argument 2 expires.
EnableHistory	BOOL	Argument 3.Bit 3: Status of the Enable bit in the previous scan.

Latch Logic Commands

Overview

The Custom Logic Editor uses the following latch commands:

- LATCH
- LATCH_NV

LATCH

The LATCH command:

- stores a Boolean value (0 or 1) in a temporary register
- provides a method for setting and clearing the stored value
- saves the clear and set status from the previous scan.

Arguments	Representation
1	`LATCH TmpReg`

Input arguments		
Argument	**Type**	**Description**
Set	BOOL	Bit 1: turns On the latch and sets the value of the State bit (Bit 0) to 1.
Clear	BOOL	Bit 2: turns Off the latch and sets the value of the State bit (Bit 0) to 0.
Output arguments:		
State	BOOL	Bit 0: the state (0 or 1) of the latch.
SetHistory	BOOL	Bit 3: contains the status of the Set bit (Bit 1) from the previous scan.
OnHistory	BOOL	Bit 4: contains the status of the Clear bit (Bit 2) from the previous scan.

LATCH_NV

The LATCH_NV command:

- stores a Boolean value (0 or 1) in a non-volatile register
- provides a method for setting and clearing the stored value
- saves the clear and set status from the previous scan.

Use the LATCH_NV command, instead of the LATCH command, to retain the latch state during a power cycle.

Arguments	Representation
1	`LATCH_NV NVReg`

Input arguments		
Argument	**Type**	**Description**
Set	BOOL	Bit 1: turns On the latch and sets the value of the State bit (Bit 0) to 1.
Clear	BOOL	Bit 2: turns Off the latch and sets the value of the State bit (Bit 0) to 0.
Output arguments:		
State	BOOL	Bit 0: the state (0 or 1) of the latch.
SetHistory	BOOL	Bit 3: contains the status of the Set bit (Bit 1) from the previous scan.
OnHistory	BOOL	Bit 4: contains the status of the Clear bit (Bit 2) from the previous scan.

Counter Logic Commands

Overview

The custom logic editor uses the following counter logic commands:

- COUNTER
- COUNTER_NV

COUNTER

The COUNTER command:

- increments or decrements a count value
- provides a method for setting the count value to a preset value
- indicates when the count value equals 0
- indicates the relationship between the count value and the preset value - equal to, greater than or less than
- saves the increment, decrement and set status from the previous scan.

Arguments	Representation
3	`COUNTER TmpReg KValue TmpReg`

Input arguments:		
Argument	**Type**	**Description**
PresetValue	UINT	Argument 2: a preset integer from 0 to 65,535. Used to: • set the count value equal to the PresetValue • compare the count value to the PresetValue
Increment	BOOL	Argument 3.Bit 4: the rising edge of this bit increases the Count by a value of 1.
Decrement	BOOL	Argument 3.Bit 5: the rising edge of this bit decreases the Count by a value of 1.
Set	BOOL	Argument 3.Bit 6: the rising edge of this bit sets the Count equal to the PresetValue.
Output arguments:		
Count	UINT	Argument 1: the count. An integer from 0 to 65,535.
Zero	BOOL	Argument 3.Bit 0: indicates that the Count value is 0.
LT	BOOL	Argument 3.Bit 1: indicates that the Count value is less than the PresetValue.
EQ	BOOL	Argument 3.Bit 2: indicates that the Count value equals the PresetValue.
GT	BOOL	Argument 3.Bit 3: indicates that the Count value is greater than the PresetValue.
IncHistory	BOOL	Argument 3.Bit 7: indicates the status of the Increment bit (Argument 3.Bit 4) from the previous scan.
DecHistory	BOOL	Argument 3.Bit 8: indicates the status of the Decrement bit (Argument 3.Bit 5) from the previous scan.
SetHistory	BOOL	Argument 3.Bit 9: indicates the status of the Set bit (Argument 3.Bit 6) from the previous scan.

COUNTER_NV

The COUNTER_NV command:

- increments or decrements a count value
- provides a method for setting the count value to a preset value
- indicates when the count value equals 0
- indicates the relationship between the count value and the preset value - equal to, greater than or less than
- saves the increment, decrement and set status from the previous scan.

Use the COUNTER_NV command, instead of the COUNTER command, to retain the count during a power cycle.

Arguments	Representation
3	`COUNTER NVReg KValue NVReg`

Input arguments:		
Argument	**Type**	**Description**
PresetValue	UINT	Argument 2: a preset integer from 0 to 65,535. Used to: ● set the count value equal to the PresetValue ● compare the count value to the PresetValue
Increment	BOOL	Argument 3.Bit 4: the rising edge of this bit increases the Count by a value of 1.
Decrement	BOOL	Argument 3.Bit 5: the rising edge of this bit decreases the Count by a value of 1.
Set	BOOL	Argument 3.Bit 6: the rising edge of this bit sets the Count equal to the PresetValue.
Output arguments:		
Count	UINT	Argument 1: the count. An integer from 0 to 65,535.
Zero	BOOL	Argument 3.Bit 0: indicates that the Count value is 0.
LT	BOOL	Argument 3.Bit 1: indicates that the Count value is less than the PresetValue.
EQ	BOOL	Argument 3.Bit 2: indicates that the Count value equals the PresetValue.
GT	BOOL	Argument 3.Bit 3: indicates that the Count value is greater than the PresetValue.
IncHistory	BOOL	Argument 3.Bit 7: indicates the status of the Increment bit (Argument 3.Bit 4) from the previous scan.
DecHistory	BOOL	Argument 3.Bit 8: indicates the status of the Decrement bit (Argument 3.Bit 5) from the previous scan.
SetHistory	BOOL	Argument 3.Bit 9: indicates the status of the Set bit (Argument 3.Bit 6) from the previous scan.

Math Logic Commands

Overview

The custom logic editor uses the following math commands:

- ON_ADD
- ON_SUB
- ON_MUL
- ON_DIV

ON_ADD

The ON_ADD command performs unsigned addition when the bit accumulator transitions from 0 to 1. It adds the value from Argument 1 to the 16-bit accumulator value, then posts the result back to the Value in Argument 1.

A status register:

- indicates an overflow if the result of the addition process exceeds 65,535, and
- indicates the status of the bit-accumulator from the previous scan.

Arguments	Representation
2	`ON_ADD TmpReg TmpReg`

Input arguments:		
Argument	**Type**	**Description**
Value	UINT	Argument 1: the amount to be added to the 16-bit accumulator. An integer from 0 to 65,535.
Output arguments:		
Value	UINT	Argument 1: the result of the addition procedure. An integer from 0 to 65,535.
Overflow	BOOL	Argument 2.Bit 0: indicates that the addition operation resulted in a value greater than 65,535. In this event, the true sum of the operation equals the value output to Argument 1 + 65,536.
OnHistory	BOOL	Argument 2.Bit 3: Status of the bit accumulator in the previous scan.

ON_SUB

The ON_SUB command performs unsigned subtraction when the bit accumulator transitions from 0 to 1. It subtracts the 16-bit accumulator value from the value in Argument 1, then posts the result back to the value in Argument 1.

A status register:

- indicates an underflow if the result of the subtraction process is less than 0, and
- indicates the status of the bit-accumulator from the previous scan.

Arguments	Representation
2	`ON_SUB TmpReg TmpReg`

Input arguments:		
Argument	**Type**	**Description**
Value	UINT	Argument 1: the amount from which the 16-bit accumulator is subtracted. An integer from 0 to 65,535.
Output arguments:		
Value	UINT	Argument 1: the result of the subtraction process. An integer from 0 to 65,535.
Underflow	BOOL	Argument 2.Bit 0: indicates that the subtraction operation resulted in a value less than 0. In this case, the true result of the operation equals the value output to Argument 1 - 65,536.
OnHistory	BOOL	Argument 2.Bit 3: Status of the bit accumulator in the previous scan.

ON_MUL

The ON_MUL command performs unsigned multiplication when the bit accumulator transitions from 0 to 1. The ON_MUL procedure multiplies the value from Argument 2 against the 16-bit accumulator value, then posts the result back to Argument 1 (most significant word) and Argument 2 (least significant word).

A status register indicates the status of the bit-accumulator from the previous scan.

Arguments	Representation
3	`ON_MUL TmpReg TmpReg TmpReg`

Input argument:		
Argument	**Type**	**Description**
Value	UINT	Argument 2: the amount to be multiplied by the value in the 16-bit accumulator. An integer from 0 to 65,535.
Output arguments:		
Product	UINT	Argument 1 & Argument 2: the result of the multiplication procedure. Argument 1 holds the most significant word; Argument_2 holds the least significant word. An integer from 0 to 65,535.
OnHistory	BOOL	Argument 3.Bit 3: Status of the bit accumulator in the previous scan.

ON_DIV

The ON_DIV command performs unsigned division when the bit accumulator transitions from 0 to 1. The ON_DIV procedure divides the combined value of Argument 1 and Argument 2 by the 16-bit accumulator value, then posts the result back to Argument 1 (most significant word) and Argument 2 (least significant word).

A status register indicates:

- an overflow if division is by 0
- the status of the bit-accumulator from the previous scan.

Arguments	**Representation**
3	`ON_DIV TmpReg TmpReg TmpReg`

Input argument:		
Argument	**Type**	**Description**
Value	UINT	Argument 2: the amount to be multiplied by the value in the 16-bit accumulator. An integer from 0 to 65,535.
Output arguments:		
Product	UINT	Argument 1 & Argument 2: the result of the division procedure. Argument 1 holds the most significant word; Argument_2 holds the least significant word. An integer from 0 to 65,535.
Overflow	BOOL	Argument 3.Bit 0: indicates division by 0.
OnHistory	BOOL	Argument 3.Bit 3: Status of the bit accumulator in the previous scan.

2.3 Structured Text Program Examples

At a Glance

Summary

This section shows the structured text program of 2 typical situations which you may need to use in your applications:

- Checking timers and multiply commands
- Creating a truth table

What's in this Section?

This section contains the following topics:

Topic	Page
How to Check Timers and Multiply Commands	98
How to Create a Truth Table	99

How to Check Timers and Multiply Commands

Overview

Customizing your application you may need to check timers and multiply commands.

Checking Timers and Multiply Commands with a Structured Text Program

The following diagram gives the structured text program in Text View of how to check timers and multiply commands

```
 LOGIC_ID 256
// A very simple test that checks timers and MUL (multiply command)
// It should switch LO1 and LO2 ON OFF if OK !!
//
 LOAD_K_BIT 1
 SET_TMP_BIT 15 3
 LOAD_TMP_REG 15
 ON_SET_TMP_REG 5 11
 ON_SET_TMP_REG 8 12
 LOAD_NOT_TMP_BIT 10 2     //  timer 2 not timing
 SET_TMP_BIT 7 0
 TIMER_TENTHS 5 6 7
 LOAD_NOT_TMP_BIT 7 2      //  timer 1 not timing
 SET_TMP_BIT 10 0
 TIMER_TENTHS 8 9 10
 LOAD_TMP_BIT 7 2
 SET_BIT 1200 12           //  Switch LO1 if timer 1 is working
 LOAD_K_REG 50             //  Load value of 50
 LOAD_K_BIT 1
 SET_NOT_TMP_BIT 23 3      //  Clear history bit
 ON_SET_TMP_REG 22 23     //  Save the 50 in temporary register 22
 LOAD_K_REG 2              //  Load value of 2
 SET_NOT_TMP_BIT 23 3
 ON_MUL 21 22 23           //  Multiply 50x2
 LOAD_TMP_REG 22
 COMP_K_REG 100 0          //  Is result 100?
 LOAD_TMP_BIT 10 2         //  timer 2 timing
 AND_TMP_BIT 0 2           //  =100?
 SET_BIT 1200 13         // Don't switch LO2 if MUL did not work OK
```

How to Create a Truth Table

Overview

Customizing your application you may need to create a truth table.

Creating a Truth Table with a Structured Text Program

The following diagram gives the structured Text program in Text View of the creation of a truth table

```
LOGIC_ID 444
//
//
// Truth table example
//
//    I1  I2  I3    Output
//     0   0   0      0     (0)
//     0   0   1      1     (1)
//     0   1   0      1     (2)
//     0   1   1      0     (3)
//     1   0   0      1     (4)
//     1   0   1      0     (5)
//     1   1   0      0     (6)
//     1   1   1      0     (7)

LOAD_BIT 516.0        //SET INPUTS
SET_TMP_BIT 1.1
LOAD_BIT 516.1
SET_TMP_BIT 1.2
LOAD_BIT 516.2
SET_TMP_BIT 1.3

//
//**** 3x1 TRUTH TABLE TEMPLATE
//**** Inputs defined as bits 1.1 through 1.3)
//**** Output defined as bit 1.15
//
LOAD_K_BIT 0             //default output OFF
SET_TMP_BIT 1.15         //save partial result

//***************0**  Inputs 1-2-3 are OFF OFF OFF
//
LOAD_NOT_TMP_BIT 1.1  //include this SECTION
AND_NOT_TMP_BIT 1.2   //if output is to be ON
AND_NOT_TMP_BIT 1.3   //REMOVE if output to be OFF
SET_TMP_BIT 1.15      //save partial result
//
```

Creating a Truth Table with a Structured Text Program (cont'd)

```
LOAD_NOT_TMP_BIT 1.1  //include this SECTION
AND_NOT_TMP_BIT 1.2   //if output is to be ON
AND_TMP_BIT 1.3       //REMOVE if output to be OFF
OR_TMP_BIT 1.15       //include previous result
SET_TMP_BIT 1.15      //save partial result
//
//***************2**  Inputs 1-2-3 are OFF ON OFF
//
LOAD_NOT_TMP_BIT 1.1  //include this SECTION
AND_TMP_BIT 1.2       //if output is to be ON
AND_NOT_TMP_BIT 1.3   //REMOVE if output to be OFF
OR_TMP_BIT 1.15       //include previous result
SET_TMP_BIT 1.15      //save partial result
//
//***************3**  Inputs 1-2-3 are OFF ON ON
//
LOAD_NOT_TMP_BIT 1.1  //include this SECTION
AND_TMP_BIT 1.2       //if output is to be ON
AND_TMP_BIT 1.3       //REMOVE if output to be OFF
OR_TMP_BIT 1.15       //include previous result
SET_TMP_BIT 1.15      //save partial result
//
//***************4**  Inputs 1-2-3 are ON OFF OFF
//
LOAD_TMP_BIT 1.1      //include this SECTION
AND_NOT_TMP_BIT 1.2   //if output is to be ON
AND_NOT_TMP_BIT 1.3   //REMOVE if output to be OFF
OR_TMP_BIT 1.15       //include previous result
SET_TMP_BIT 1.15      //save partial result
//
//***************5**  Inputs 1-2-3 are ON OFF ON
//
LOAD_TMP_BIT 1.1      //include this SECTION
AND_NOT_TMP_BIT 1.2   //if output is to be ON
AND_TMP_BIT 1.3       //REMOVE if output to be OFF
OR_TMP_BIT 1.15       //include previous result
SET_TMP_BIT 1.15      //save partial result
```

Creating a Truth Table with a Structured Text Program (cont'd)

```
//
//***************6**  Inputs 1-2-3 are ON ON OFF
//
LOAD_TMP_BIT 1.1      //include this SECTION
AND_TMP_BIT 1.2       //if output is to be ON
AND_NOT_TMP_BIT 1.3   //REMOVE if output to be OFF
OR_TMP_BIT 1.15       //include previous result
SET_TMP_BIT 1.15      //save partial result
//
//***************7**  Inputs 1-2-3 are ON ON ON
//
LOAD_TMP_BIT 1.1      //include this SECTION
AND_TMP_BIT 1.2       //if output is to be ON
AND_TMP_BIT 1.3       //REMOVE if output to be OFF
OR_TMP_BIT 1.15       //include previous result
SET_TMP_BIT 1.15      //save partial result


LOAD_TMP_BIT 1.15     //SET OUTPUT
SET_BIT 517.3
```

2.4 Compiling and Simulation of a Structured Text Language Program

At a Glance

Summary

The following section describes how to compile a structured text language program. It also describes the user interface windows involved in the compiling of the program and the logic simulator.

What's in this Section?

This section contains the following topics:

Topic	Page
Introduction	103
PCode Window	104
Error Window	106
Output Window	108
Quick Watch Window	110
LTM R Controller Logic Simulator	112

Introduction

Compiling Overview

You must compile the structured text language program before you can download it to the LTM R controller.

Compiling includes a check for program errors, such as:

- syntax and structure errors
- symbols without corresponding addresses
- resources used by the program that are not available
- whether the program fits in available controller memory

Compiling a Program

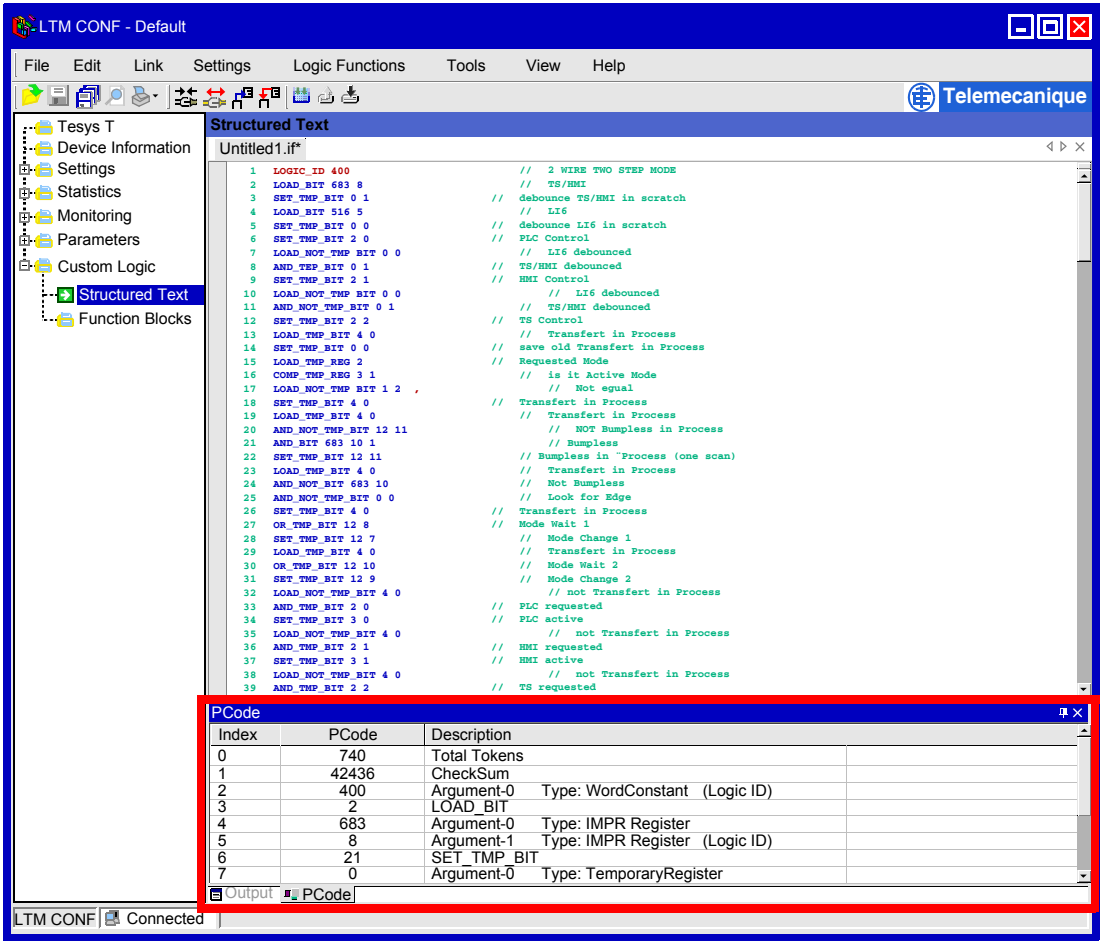
Once you finish editing the program, follow these steps to compile it:

Step	Action
1	Click Logic Functions in the top-level menu bar.
2	Click Compile in the displayed window. If no errors are detected, the PCode window is displayed. Otherwise, the Errors window is displayed.

PCode Window

Overview

When a custom logic program is compiled successfully, the PCode (Pseudo Code) window is displayed:


LTM CONF - Default
File Edit Link Settings Logic Functions Tools View Help
Telemecanique
Tesys T
Device Information
Settings
Statistics
Monitoring
Parameters
Custom Logic
Structured Text
Function Blocks
Structured Text
Untitled1.if*
1 LOGIC_ID 400 // 2 WIRE TWO STEP MODE
2 LOAD_BIT 683 8 // TS/HMI
3 SET_TMP_BIT 0 1 // debounce TS/HMI in scratch
4 LOAD_BIT 516 5 // LI6
5 SET_TMP_BIT 0 0 // debounce LI6 in scratch
6 SET_TMP_BIT 2 0 // PLC Control
7 LOAD_NOT_TMP_BIT 0 0 // LI6 debounced
8 AND_TEP_BIT 0 1 // TS/HMI debounced
9 SET_TMP_BIT 2 1 // HMI Control
10 LOAD_NOT_TMP_BIT 0 0 // LI6 debounced
11 AND_NOT_TMP_BIT 0 1 // TS/HMI debounced
12 SET_TMP_BIT 2 2 // TS Control
13 LOAD_TMP_BIT 4 0 // Transfert in Process
14 SET_TMP_BIT 0 0 // save old Transfert in Process
15 LOAD_TMP_REG 2 // Requested Mode
16 COMP_TMP_REG 3 1 // is it Active Mode
17 LOAD_NOT_TMP_BIT 1 2 // Not egual
18 SET_TMP_BIT 4 0 // Transfert in Process
19 LOAD_TMP_BIT 4 0 // Transfert in Process
20 AND_NOT_TMP_BIT 12 11 // NOT Bumpless in Process
21 AND_BIT 683 10 1 // Bumpless
22 SET_TMP_BIT 12 11 // Bumpless in Process (one scan)
23 LOAD_TMP_BIT 4 0 // Transfert in Process
24 AND_NOT_BIT 683 10 // Not Bumpless
25 AND_NOT_TMP_BIT 0 0 // Look for Edge
26 SET_TMP_BIT 4 0 // Transfert in Process
27 OR_TMP_BIT 12 8 // Mode Wait 1
28 SET_TMP_BIT 12 7 // Mode Change 1
29 LOAD_TMP_BIT 4 0 // Transfert in Process
30 OR_TMP_BIT 12 10 // Mode Wait 2
31 SET_TMP_BIT 12 9 // Mode Change 2
32 LOAD_NOT_TMP_BIT 4 0 // not Transfert in Process
33 AND_TMP_BIT 2 0 // PLC requested
34 SET_TMP_BIT 3 0 // PLC active
35 LOAD_NOT_TMP_BIT 4 0 // not Transfert in Process
36 AND_TMP_BIT 2 1 // HMI requested
37 SET_TMP_BIT 3 1 // HMI active
38 LOAD_NOT_TMP_BIT 4 0 // not Transfert in Process
39 AND_TMP_BIT 2 2 // TS requested
PCode
Index PCode Description
0 740 Total Tokens
1 42436 CheckSum
2 400 Argument-0 Type: WordConstant (Logic ID)
3 2 LOAD_BIT
4 683 Argument-0 Type: IMPR Register
5 8 Argument-1 Type: IMPR Register (Logic ID)
6 21 SET_TMP_BIT
7 0 Argument-0 Type: TemporaryRegister
Output
PCode
LTM CONF
Connected

PCode Window Elements

The following table lists the different elements which make up the PCode window:

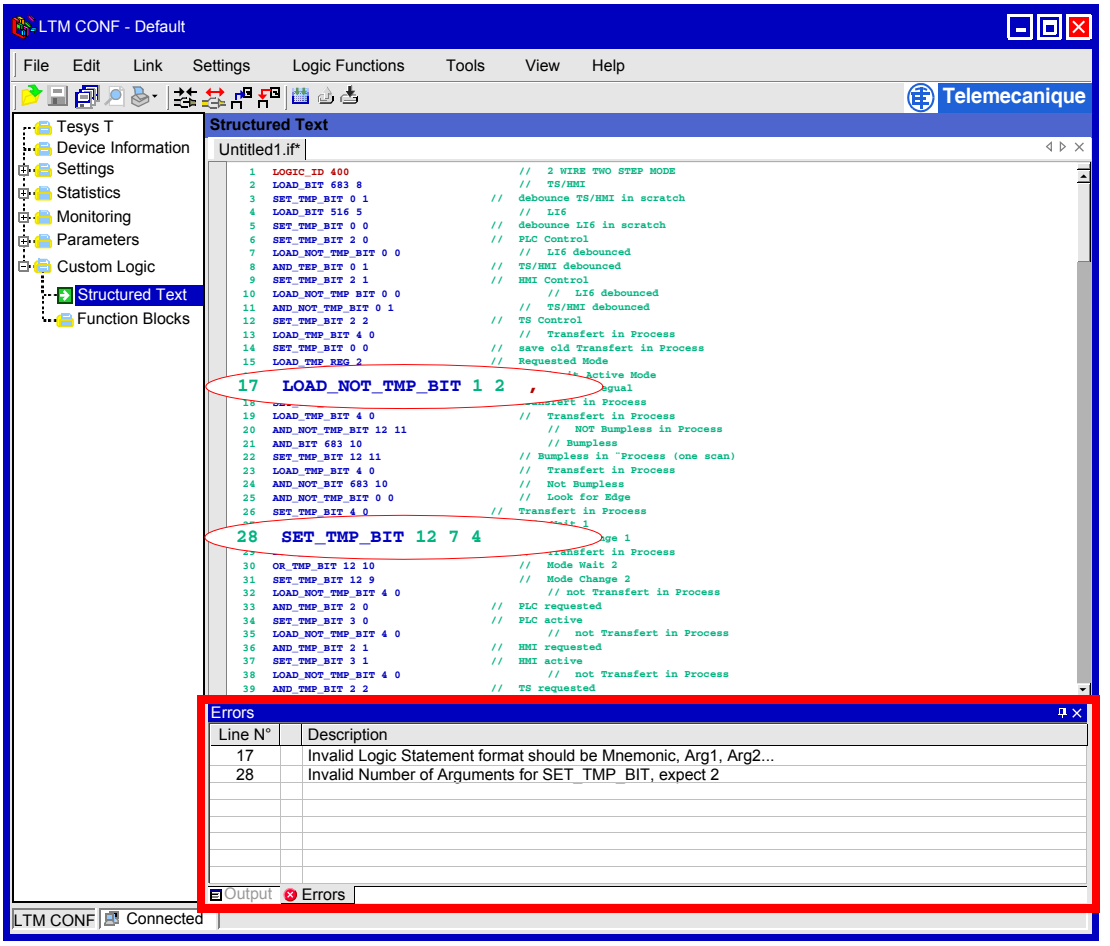
Item	Description
Total tokens	Size of PCode (in 16 bits word). Count including checksum, logic ID and all logic commands and arguments
Checksum	Module 16 summation of all logic commands and arguments.
Logic Command	Each logic command in the program and its related Pcode.
Argument	Each argument in the program, and the type of register (temporary, non-volatile or data) that it refers to or affects.

Note: Logic commands and arguments are listed in the same order as in the structured text language program.

Error Window

Overview

When a structured text language program is compiled, it may contain errors. In this case, the Error window is displayed:



Error Window Elements

In the example above, 2 mistakes were made.

The Error window indicates:

- the line numbers with errors, and
- a description of the error.

Error Types

The following list describes the different types of errors that may occur:

- syntax and structure errors
- logic commands without corresponding addresses
- resources used by the program that are not available
- program size is too big

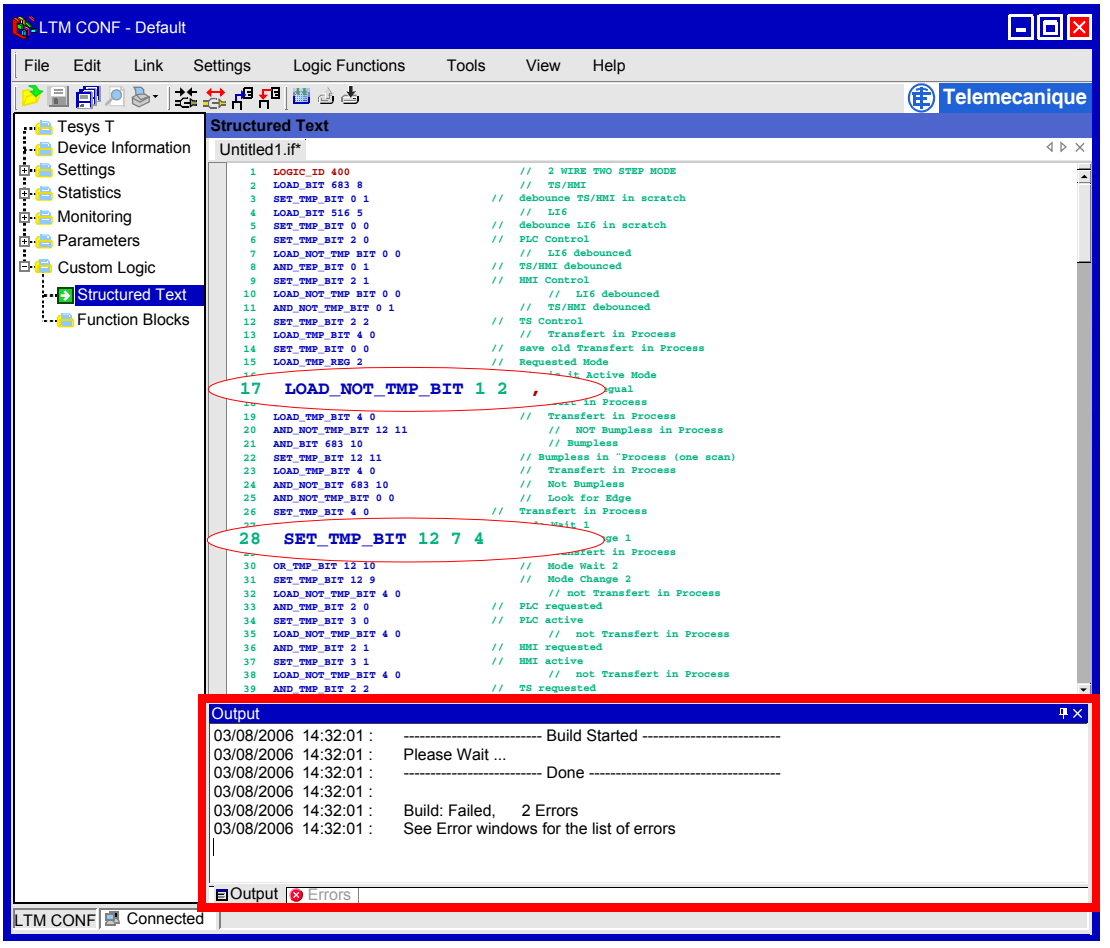
Output Window

Overview

You can access the Output window from either the Error or the PCode window.

To do this, click the tab on the left of the Error or PCode tab at the very bottom of the screen.

The following illustration shows the Output window:



Output Window Elements

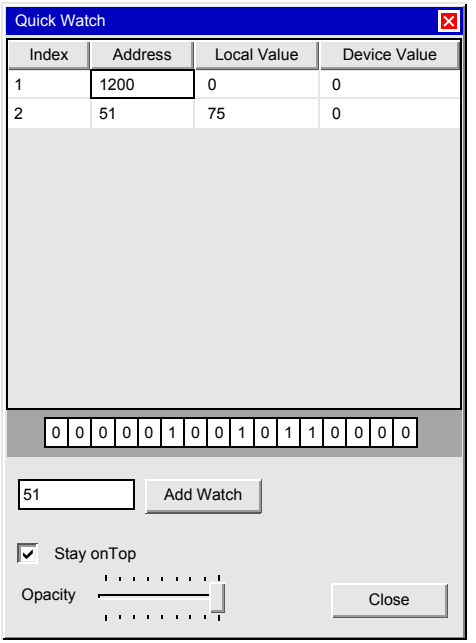
The Output window shows the logs associated with the compilation of the structured text language program and indicates whether it was successful.

In the example above, 2 errors were encountered, so you are prompted to view the Errors window. You can access the Error window by clicking the tab on the right of the Output tab at the very bottom of the screen or select View -> Error Window in the top level menu bar.

Quick Watch Window

Overview

The Quick Watch window enables you to easily monitor the registers you select:



Adding a Register in the Quick Watch Window

The following table explains how to add a register in the Quick Watch window:

Step	Action
1	Click **View** in the top level menu bar.
2	Click on **Quick Watch Window**. **Result:** The Quick Watch window opens.
3	Type a register number in the box on the left of the **Add Watch** button
4	Click **Add Watch**. **Result:** The register is added, and its local and device values are shown.
5	Repeat step 3 and 4 for every register you want to add to the list.

Register Value

Click a number in one of the boxes to display its value in binary code in the middle of the Quick Watch window.

For example, in the figure above, if you click 1200 (the number of the register that was added) it will be displayed in binary code.

Quick Watch Window Functions

The following functions are also available in the Quick Watch window

Functionality	Description
Stay on Top	When active, the Quick Watch window remains in the foreground regardless of which application is active. It enables you to navigate in the software or in another program while still monitoring the added registers.
Opacity	Enables you to adjust the opacity of the Quick Watch window.

Index

Registers you choose to monitor are appended to the Quick Watch window. An index number is assigned to each register.

Local Value

The local value corresponds to the value that LTM CONF configuration software assigns to this register.

Device Value

The device value corresponds to the value which is stored in the LTM R controller. The Quick Watch window displays actual "Device Values" only while a Parameters view window (i.e. All Parameters, Configurable Parameters..) is open.

LTM R Controller Logic Simulator

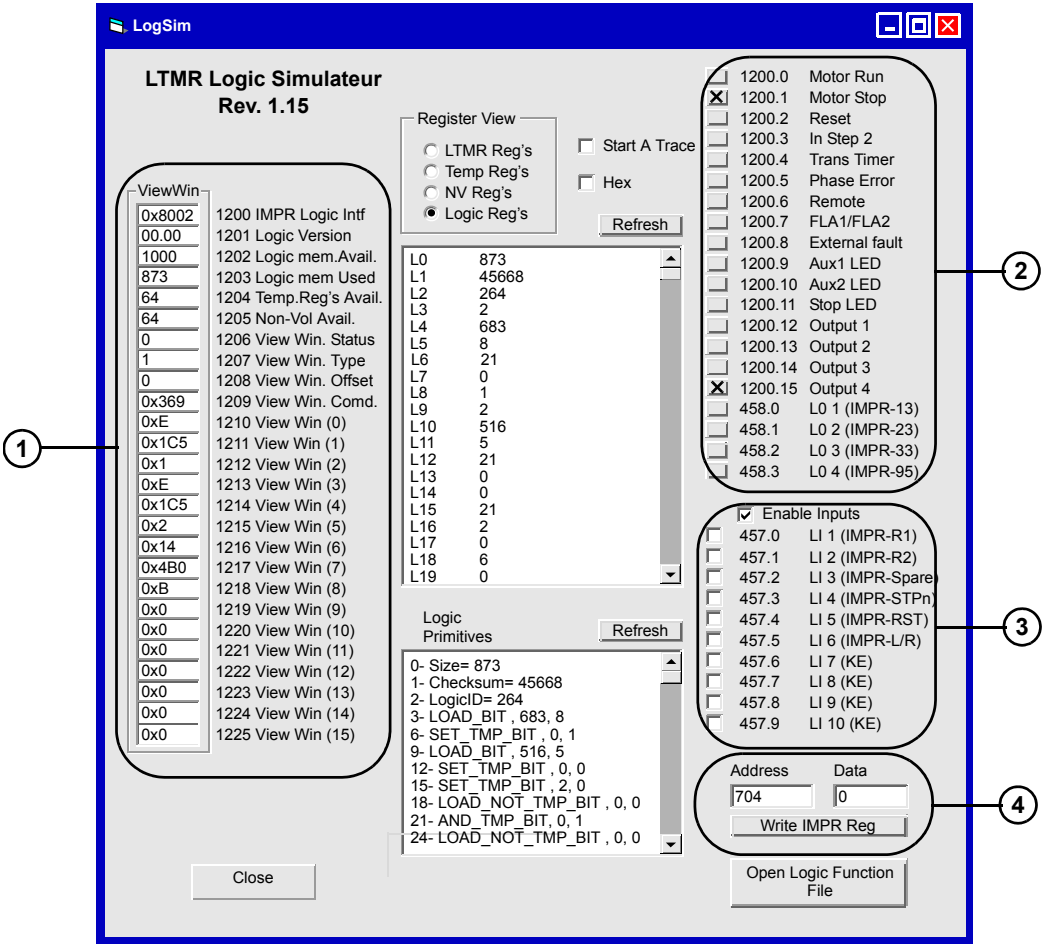
Overview

LTM CONF programming software comes with the LTM R controller logic simulator. It enable to test the functionning of a custom logic program before transferring it into the LTM R controller.

Logic Simulator Interface

To open the logic simulator, navigate in the top-level menu bar to Tools → Logic Simulator. The logic simulator is then displayed. In the right bottom corner (see below), click on the 'Load *.lf File' button to import your program you previously compiled.

The logic simulator with a loaded custom logic file is then displayed:



Register View

4 kinds of registers are displayed by the logic simulator

- LTM R controller registers.
- Temporary registers.
- Non-volatile registers
- Logic memory

Those registers can not be displayed in the same time. The Register View enables you to choose which ones you wish to monitor. In the example above, the content of the logic memory is displayed.

Note: By default, registers values are displayed in decimal code. Tick the "Hex" box if you would prefer them to be in hexadecimal code.

Logic Primitives Window

The Logic Primitives window displays the compiled PCode (see *p. 104*).

Note: The PCode may read or write to any READ/WRITE register that is accessible by serial port communications.

View Window

The logic simulator displays the content of LTM R controller registers 1200 to 1225 in hexadecimal code (See part 1 on the illustration above). Registers 1200 to 1205 (see p. 26) are the custom logic registers.

Registers 1200 and 458

The logic simulator displays the status of registers 1200 and 458 (See part 2 on the illustration above). The LTM R controller firmware then reads those PCode register values to direct device functions and physical outputs. For more information about those registers see the sections on Communication Variables in the *Use* chapter of the *Motor Management Controller :TeSys® T LTM R User's Manual.*

The logic simulator displays an X in each output status checkbox to indicate that a bit value of '1' exists in the output status register.

Register 457

The logic simulator enables to write values to register 457 bits (see part 3 on the illustration above) .For more information about this register see the sections on Communication Variables in the *Use* chapter of the *Motor Management Controller :TeSys® T LTM R User's Manual*. To be allowed to write to register 457, tick the 'Enable Inputs' box.

Ticking a box on the left of a register bit will assign a value of 1 to this bit. Untick this box to assign 0 to this bit.

Example: If you tick the 3 first boxes, bits 457.0, 457.1 and 457.2 will get the value of 1. Click on the upper refresh button, and then check the value of register 457. You can see that it has the value of 7, which is in binary code 0000000000000111.

Writing to a Register address

The logic simulator enables to write data to any register address (see part 4 on the illustration above). Follow these steps to assign a value to a register:

Step	Action
1	Untick the 'Enable Inputs' box.
2	Specify to which register you wish write data in the 'Address' box.
3	Specify which value you wish to assign in the 'Data' box.
4	Click on the 'Write IMPR Reg' button.

Start a Trace

The 'Start a Trace' box is an integrated debugging tool which captures the 1-bit and the 16-bit accumulator content

Refresh

When you load your *.lf file into the logic simulator, it emulates the LTM R controller behavior. However, values are assigned when you load the file, regardless of changes you made in the logic simulator. Click the upper refresh button to take into acounts the changes made to registers' values . Click the bottom refresh button to refresh the PCode.

Function Block Diagram Language

3

At a Glance

Overview

This chapter describes how to use the FBD (Function Block Diagram) programming language with LTM CONF programming software.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
3.1	Overview of FBD Language	116
3.2	FBD Elements	121
3.3	Programming with the FBD Language	137
3.4	Manipulating FBD Blocks	144
3.5	FBD Editor Display Options	147

3.1 Overview of FBD Language

At a Glance

Summary

This section provides a general description of FBD language. Use the FBD language to customize a pre-defined operating mode or to create a new program to suit the requirements of a specific application created using FBD.

What's in this Section?

This section contains the following topics:

Topic	Page
Introduction to the FBD Editor	117
FBD Editor Toolbox	120

Introduction to the FBD Editor

Overview

The FBD editor is a feature of TeSys® configuration software. Use the FBD editor to view an existing FBD file or to create a new FBD file using FBD language, rather than an instruction-based text programming language.

Creating an FBD Program

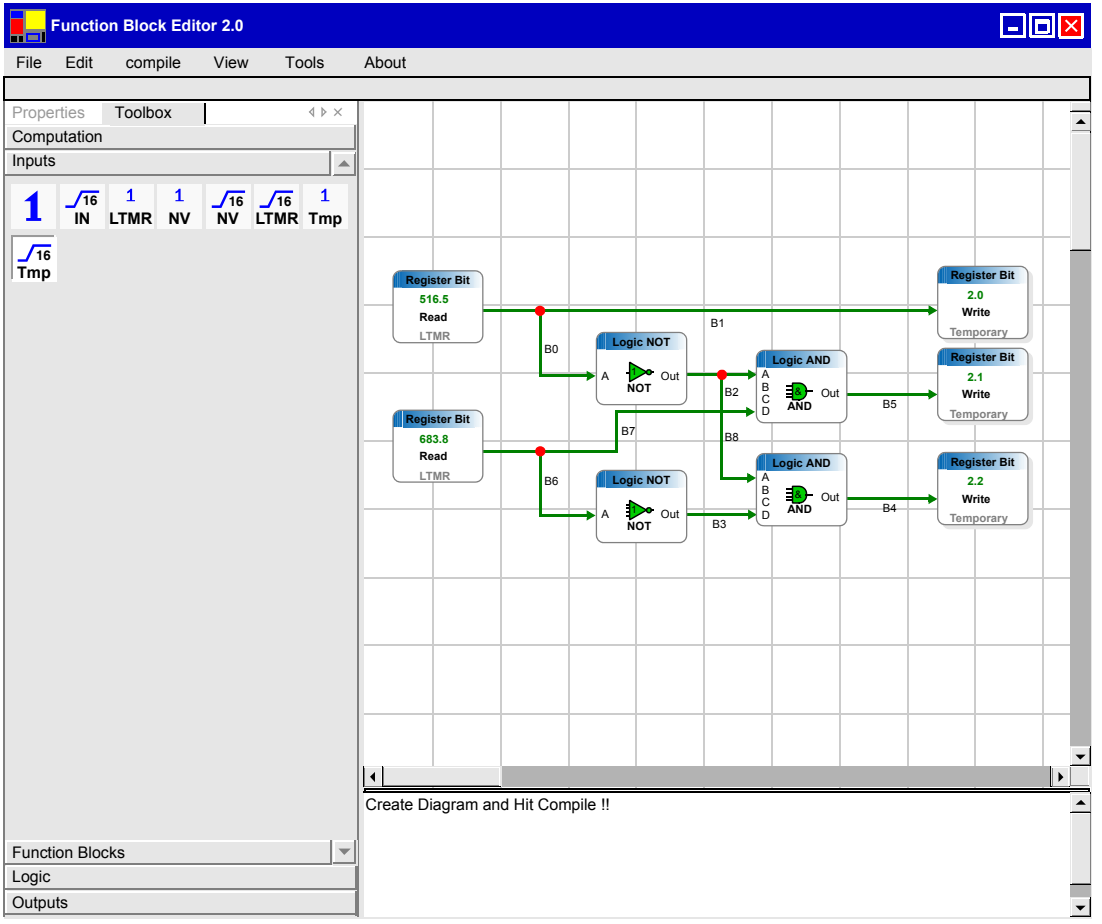
To open the FBD editor, navigate in the configuration software tree (on the left of the screen) to Custom Logic → Function Blocks. This will open the FBD editor in the main window.

Each FBD file, when you save it, has a file extension of ".Gef".

FBD Editor User Interface

The FBD editor is available even when the configuration software is not connected to the LTM R controller. However, many of the menu items will be enabled only when an FBD program is open in the FBD editor.

When an FBD file is open the FBD editor looks like this:



Workspace

FBD programs are edited and created in the workspace.

The workspace is made up of 2 elements:

- blocks, and
- wires to link the blocks.

How to Create and Use a FBD Program

Follow these steps to create an and use an FBD program:

Step	Action
1	Select Settings → Motor → Motor Operating Mode to use the custom logic editor in the configuration software.
2	Set the operating mode to **Custom**.
3	Navigate in the configuration software tree (on the left of the screen) to Custom Logic → Function Blocks. **Result**: The FBD editor opens.
4	Create your FBD program using blocks and link them with wires in the workspace.
5	Use the **Compile to structured text** command in the FBD editor's **Compile** menu to validate the FBD file when you have finished editing it. **Result**: The FBD program compiles to structured text.
6	Minimize the FBD editor window and navigate in the configuration software tree (on the left of the screen in LTM CONF configuration software) to Custom Logic → Structured Text. **Result**: The program is automatically copied into the structured text editor and has a *.glf extension.
7	Compile this program by clicking **Compile** in the menu displayed in the LTM CONF main toolbar when the structured text editor is open. **Result**: The PCode (Pseudo Code) window notifies you when the logic file has successfully compiled. The PCode window is a feature of LTM CONF programming software.
8	Download the completed logic file from the custom logic editor to the LTM R controller using the logic functions menu **Download Program to Device** command.

FBD Editor Toolbox

At a Glance

To create an FBD program, the different functions to be inserted in the workspace are available in the toolbox. Each of the tabs in the toolbox groups a function type. When you click one of the tabs, it displays the list of available blocks.

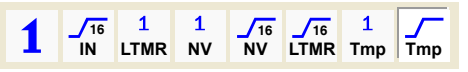
Note: You can navigate between the different tabs holding the Ctrl key and pressing the Tab key.

Computation Blocks

The following figure shows the Computation blocks:

Inputs Blocks

The following figure shows the Inputs blocks:

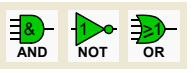


Function Blocks

The following figure shows the Function blocks:

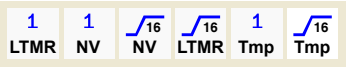
Logic Blocks

The following figure shows the Logic blocks:



Outputs Blocks

The following figure shows the Outputs blocks:



3.2 FBD Elements

FBD Elements Overview

Summary

This section describes in detail the FBD elements provided by the FBD editor, and their inputs/outputs.

What's in this Section?

This section contains the following topics:

Topic	Page
Computation Blocks	122
Inputs Blocks	125
Function Blocks	128
Logic Blocks	133
Outputs Blocks	134

Computation Blocks

Overview

The FBD editor uses various Computation blocks:

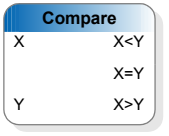
- Compare
- Add
- Division
- Multiplication
- Substraction

Access

To access computation blocks, click on the Computation bar in the Toolbox. This is then displayed in the Toolbox:

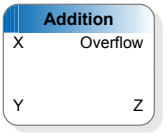
Compare Block Characteristics

The block compares two 16-bit register values. The following table describes the Compare block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	X: 16-bit unsigned register value (0 to 65,535). Y: 16-bit unsigned register value (0 to 65,535).
	Outputs	X < Y: ON/OFF (0 or 1) temporary bit that is ON if the value X is less than the value Y. X = Y: ON/OFF (0 or 1) temporary bit that is ON if the value X is equal to the value Y. X > Y: ON/OFF (0 or 1) temporary bit that is ON if the value X is greater than the value Y.

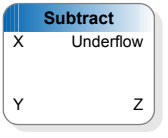
Add Block Characteristics

The block performs an unsigned addition of two 16-bit register values. The following table describes the Add block characteristics:

FBD symbol	Inputs/outputs or example	Description
	Inputs	X: 16-bit unsigned register value (0 to 65,535). Y: 16-bit unsigned register value (0 to 65,535).
	Outputs	Z : 16-bit unsigned register result (Z = X + Y). Overflow: ON or OFF value which when set ON caries a value of 65,536. The value is initialized to OFF.
	Example	Assuming X = 60,000 and Y = 7,000, the overflow will be ON because 60,000 + 7,000 = 67,000, which is superior to 65,536. The result Z is then equal to 1,464 (1,464 + 65,356 = 67,000).

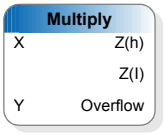
Subtraction Block Characteristics

The block performs an unsigned subtraction of two 16-bit register values. The following table describes the Subtraction block characteristics:

FBD symbol	Inputs/outputs or example	Description
	Inputs	X: 16-bit unsigned register value (0 to 65,535). Y: 16-bit unsigned register value (0 to 65,535).
	Outputs	Z : 16-bit unsigned register result (Z = X-Y). Underflow: ON or OFF value, which when set ON, caries a value of negative 65,536. The value is initialized to OFF.
	Example	Assuming X = 5 and Y = 10, the underflow will be ON because the result is negative. The result Z is then equal to 65,531 (65,531 - 65,536 = -5)

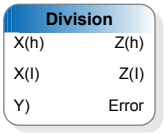
Multiplication Block Characteristics

The block performs an unsigned multiplication of two 16-bit register values. The following table describes the Multiplication block characteristics:

FBD symbol	Inputs/outputs or example	Description
	Inputs	X: 16-bit unsigned register value (0 to 65,535) Y: 16-bit unsigned register value (0 to 65,535)
	Outputs	Z(h): 16 most significant bits of the 32-bit product (Z(h) = (X * Y) / 65,536) Z(l): 16 least significant bits of the 32-bit product (Z(l) = (X * Y - Z(h)) / 65,536))
	Example	Assuming X = 20,000 and Y = 10, Z(h) = 3 and Z(l)= 3,392 because 200,000 = 3 * 65,536 + 3392

Division Block Characteristics

The block performs an unsigned division of two 16-bit register values. The following table describes the Division block characteristics:

FBD symbol	Inputs/outputs or example	Description
	Inputs	X(h): 16 most significant bits of an unsigned register value (0 to 65,535). X(l): 16 least significant bits of an unsigned register value (0 to 65,535). Y: 16-bit unsigned register divisor (0 to 65,535).
	Outputs	Z(h): 16 most significant bits of the 32-bit quotient (Z(h) = (X / Y) / 65,536) Z(l): 16 least significant bits of the 32-bit quotient (Z(l) = (X / Y - Z(h)) / 65,536)) Error : ON or OFF value, which is set ON when a division by zero occurs. This value is initialized to OFF.
	Example	Assuming X(h) = 3, X(l) = 3,392 and Y=40, Z(h) = 0 and Z(l) = 5,000 because X(h) * 65,536 + X(l) = 3 * 65,536 + 3392 = and 200,000 /Y =5,000 = 0 * 65,536 + 5,000

Inputs Blocks

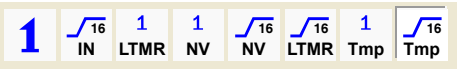
Overview

The FBD editor uses various inputs blocks:

- Constant Bit
- Constant Word
- Register Bit In
- Register Word In
- Register NV Bit In
- Register NV Word In
- Register Temp Bit In
- Register Temp Word In

Access

To access Inputs blocks, click on the Inputs bar in the Toolbox. This is then displayed in the Toolbox:



Constant Bit Block

The

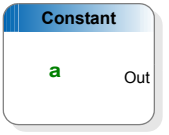
block is used to set other blocks' inputs to 0 or 1. The following table describes the Constant Bit block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	a: Constant bit value 0 or 1 (ON=1 and OFF=0).
	Outputs	Constant value 0 or 1 (ON=1 and OFF=0).

Constant Word Block

The

block is used to set other blocks' inputs values. The following table describes the Constant Word block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	a: Constant register value from 0 to 65,535.
	Outputs	Constant register value from 0 to 65,535.

Register Bit In Block

The

block enables the reading and use of a register bit value from the LTM R controller R/W addresses 0 to 1399. The following table describes the Register Bit In block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	a: Any LTM R controller register from 0 to 1399 that can be accessed via serial communication ports. b: Bit position from 0 to 15.
	Outputs	Value 0 or 1 (ON=1 and OFF=0).

Register Word In Block

The

block enables the reading and use of a register value from the LTM R controller R/W addresses 0 to 1399. The following table describes the Register Word In block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	a: Any LTM R controller register from 0 to 1399 that can be accessed via serial communication ports.
	Outputs	Value from 0 to 65,535.

Register NV Bit In Block

The

block enables the reading and use of a non-volatile register bit value. The following table describes the Register NV Bit In block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	a: Any non-volatile register from 0 to 63 b: Bit position from 0 to 15.
	Outputs	Value 0 or 1 (ON=1 and OFF=0).

Register NV Word In Block

The

block enables the reading and use of a non-volatile register value. The following table describes the Register NV Word In block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	a: Any non-volatile register from 0 to 63.
	Outputs	Value from 0 to 65,535.

Register Temp Bit In Block

The

block enables the reading and use of a temporary register bit value. The following table describes the Register Temp Bit In block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	a: Any temporary register from 0 to 299. b: Bit position from 0 to 15.
	Outputs	Value 0 or 1 (ON=1 and OFF=0).

Register Temp Word In Block

The

block enables the reading and use of a temporary register value. The following table describes the Register Temp Word In block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	a: Any temporary register from 0 to 299.
	Outputs	Value from 0 to 65,535.

Function Blocks

Overview

The FBD editor uses various Function blocks:

- Counter
- Counter NV
- Volatile Latch
- Non-volatile Latch
- Mux
- TimerSeconds
- TimerTenthSeconds

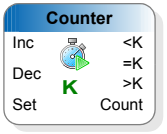
Access

To access Function blocks, click on the Function blocks bar in the Toolbox. The following menu is then displayed in the Toolbox:

Note: Placing cursor over the icon will reveal a tool tip defining the icon. This will help you distinguish which type of counter, latch, mux or timer is represented by that icon.

Counter Block Characteristics

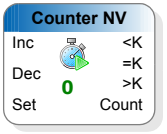
The function performs a comparative count, saving both the count and status to temporary registers. The following table describes the Counter block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	K: 16-bit unsigned constant (0 to 65,535), which specifies a preset count value. Inc: ON/OFF input value. The counter value increments by one when this input transitions from OFF to ON. Count value shall "roll over" from 65,535 to 0. Dec: ON/OFF input value. The counter value decrements by one when this input transitions from OFF to ON. Count value shall "roll over" from 0 to 65,535. Set: ON/OFF input value. The count value is set to preset value when this input transitions from OFF to ON.
	Outputs	Count: 16-bit unsigned counter value (0 to 65,535). Count is initialized to zero on power-up. <K: ON/OFF temporary bit which is ON if counter value is less than K. =K: ON/OFF temporary bit which is ON if counter value is equal to K. >K: ON/OFF temporary bit which is ON if counter value is greater than K.

Note: The Counter block range is 0 to 65,535. Cascading timers and compare functions can be used if you need larger values or multiple thresholds.

Counter NV Block Characteristics

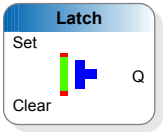
The function performs a comparative count, saving both the count and status to non-volatile registers. The following table describes the Counter NV block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	K: 16-bit unsigned constant (0 to 65,535), which specifies a preset count value. Inc: ON/OFF input value. The counter value increments by one when this input transitions from OFF to ON. Count value shall "roll over" from 65,535 to 0. Dec: ON/OFF input value. The counter value decrements by one when this input transitions from OFF to ON. Count value shall "roll over" from 0 to 65,535. Set: ON/OFF input value. The count value is set to preset value when this input transitions from OFF to ON.
	Outputs	Count: 16-bit unsigned counter value (0 to 65,535). This value is saved in non-volatile memory and initialized to the previous value on power-up. <K: ON/OFF temporary bit which is ON if counter value is less than K. =K: ON/OFF temporary bit which is ON if counter value is equal to K. >K: ON/OFF temporary bit which is ON if counter value is greater than K.

Note: The Counter block range is 0 to 65,535. Cascading timers and compare functions can be used if you need larger values or multiple thresholds

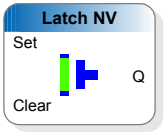
Volatile Latch Block Characteristics

The function records and retains signal history in a temporary register. The following table describes the Volatile Latch block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	Set: ON/OFF input value. The latch value is set ON when this input transitions from OFF to ON. Clear: ON/OFF input value. The latch value is set OFF when this input transitions from OFF to ON.
	Outputs	Q: ON or OFF latch value which represents the state of this latch. This value remains ON/OFF until the next rising edge of Set or Clear. This value is initialized to OFF.

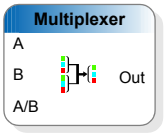
Non-Volatile Latch Block Characteristics

The function records and retains signal history in a non-volatile register. The following table describes the Non-Volatile Latch block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	Set: ON/OFF input value. The latch value is set ON when this input transitions from OFF to ON. Clear: ON/OFF input value. The latch value is set OFF when this input transitions from OFF to ON.
	Outputs	Q: ON or OFF non-volatile register bit value that represents the state of this latch. This value remains ON/OFF until the next rising edge of Set or Clear. This value is saved in non-volatile memory and initialized to previous state on power-up.

Mux Block Characteristics

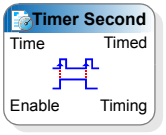
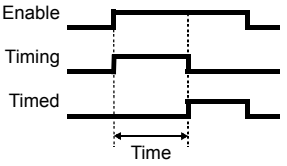
The function enables you to choose between two 16-bit unsigned values. The following table describes the Mux block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	A: 16-bit unsigned value (0 to 65,535). B: 16-bit unsigned value (0 to 65,535). A/B: ON/OFF (0 or 1) input value that selects value A or B.
	Outputs	Out: Selected 16-bit value. If A/B is OFF then Out = A. If A/B is ON then Out = B.

Timer Seconds Block Characteristics

The

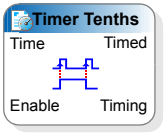
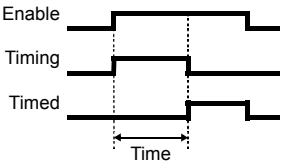
function measures time in intervals of seconds. The following table describes the Timer Seconds block characteristics:

FBD symbol	Timing diagram	Inputs/ outputs	Description
		Inputs	Time: 16-bit unsigned value (0 to 65,535) that specifies time period in seconds. Enable: ON/OFF input value. The time period is loaded on the rising edge of the "Enable" input. Time measuring continues while "Enable" is ON. Timing stops and outputs are OFF when "Enable" is OFF.
		Outputs	Timed: ON/OFF value which turns ON while "Enable" is ON and time period expires. It is OFF while measuring time or while "Enable" is OFF. Timing: ON/OFF value that is ON while "Enable" is ON and while measuring time. It is OFF after time period expires or "Enable" is OFF. Note: Both outputs can never be simultaneously ON.

Timer TenthSeconds Block Characteristics

The

function measures time in intervals of seconds. The following table describes the Timer TenthsSeconds block characteristics:

FBD symbol	Timing diagram	Inputs/ outputs	Description
		Inputs	Time: 16-bit unsigned value (0 to 65,535) that specifies time periods in tenths of seconds. Enable: ON/OFF input value. The time period is loaded on the rising edge of the "Enable" input. Time measuring continues while "Enable" is ON. Timing stops and outputs are OFF when "Enable" is OFF.
		Outputs	Timed: ON/OFF value that turns ON while "Enable" is ON and time period expires. It is OFF while measuring time or while "Enable" is OFF. Timing: ON/OFF value which is ON while "Enable" is ON and while measuring time. It is OFF after time period expires or "Enable" is OFF. Note: Both outputs can never be simultaneously ON.

Logic Blocks

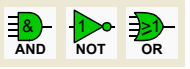
Overview

The FBD editor uses various Logic blocks:

- AND
- NOT
- OR

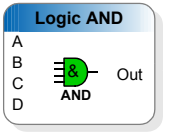
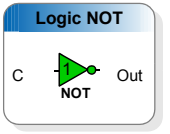
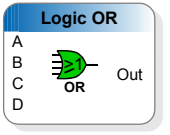
Access

To access Logic blocks, click on the Logic blocks bar in the Toolbox. This menu is then displayed in the Toolbox:



Logic Functions Blocks

The following table shows the various Logic functions :

Function	Symbol in the Toolbox	Symbol in the workspace	Description
AND			If all the inputs (ON or OFF values, respectively 1 or 0) are active (ON) or not connected, the output is active. If at least one input is inactive, the output is inactive. Note: unconnected inputs are assumed to be ON.
NOT			If the input (ON or OFF values, respectively 1 or 0) is ON, the output is OFF. If the input is OFF , the output is ON.
OR			If at least one input (ON or OFF values, respectively 1 or 0) is ON, the output is ON. If all the inputs are OFF or not connected, the output is OFF. Note: unconnected inputs are assumed to be OFF.

Outputs Blocks

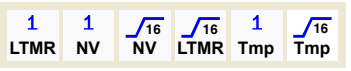
Overview

The FBD editor uses various Outputs blocks:

- Register Bit Out
- Register Word Out
- Register NV Bit Out
- Register NV Word Out
- Register Temp Bit Out
- Register Temp Word Out

Access

To access Outputs blocks, click on the Outputs bar in the Toolbox. This menu is then displayed in the Toolbox:



Register Bit Out Block

The

block is used to set an LTM R controller register bit value to 0 or 1 from the LTM R controller R/W addresses 0 to 1399. The following table describes the Register Bit Out block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	0 or 1 (ON=1 and OFF=0)
	Outputs	a: Any LTM R controller register value from 0 to 1399, which can be written via serial communication ports. b: Bit position from 0 to 15.

Register Word Out Block

The

block is used to set an LTM R controller register value from the LTM R controller R/W addresses 0 to 1399. The following table describes the Register Word Out block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	16-bit unsigned value from 0 to 65,535
	Outputs	a: Any LTM R controller register from 0 to 1399 which can be written via serial communication ports.

Register NV Bit Out Block

The

block is used to set a non-volatile register bit value to 0 or 1. The following table describes the Register NV Bit Out block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	0 or 1 (ON=1 and OFF=0)
	Outputs	a: Any non-volatile register from 0 to 63. b: Bit position from 0 to 15.

Register NV Word Out Block

The

block is used to set a non-volatile register value. The following table describes the Register NV Word Out block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	16-bit unsigned value from 0 to 65,535
	Outputs	a: Any non-volatile register from 0 to 63.

Register Temp Bit Out Block

The

block is used to set a temporary register bit value to 0 or 1. The following table describes the Register Temp Bit Out block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	0 or 1 (ON=1 and OFF=0)
	Outputs	a: Any temporary register from 0 to 299. b: Bit position from 0 to 15.

Register Temp Word Out Block

The

block is used to set a temporary register value. The following table describes the Register Temp Word Out block characteristics:

FBD symbol	Inputs/outputs	Description
	Inputs	16-bit unsigned value from 0 to 65,535
	Outputs	a: Any temporary register from 0 to 299.

3.3 Programming with the FBD Language

At a Glance

Summary

This section describes how to create and modify a program using the FBD language.

What's in this Section?

This section contains the following topics:

Topic	Page
Inserting FBD Blocks	138
Creation of Links between Blocks	139
FBD Blocks Properties	141
FBD Resource Management	142
Compiling an FBD program	143

Inserting FBD Blocks

At a Glance

To create an FBD program, you must insert blocks into the workspace, then link them together. All types of blocks can be placed in the workspace.

Inserting Blocks from the Toolbox

The following procedure describes how to insert a block from the toolbox into the workspace:

Step	Action
1	Select View -> Toolbox or left-click on the toolbox tab in the upper left corner.
2	Select the type of block to insert: • Computation • Inputs • Function Blocks • Logic • Outputs
3	Left-click on the icon corresponding to the block to insert.
4	Drag and drop the block from the toolbox to the workspace.
5	Position the block in the required location on the workspace.
6	Repeat steps 2 to 5 to insert all the blocks required for the program.

Inserting Blocks from the Workspace

The following procedure describes how to insert a block directly from the workspace:

Step	Action
1	Right-click anywhere on a blank space in the workspace. **Result**: A menu opens and enables you to choose the type of block you wish to insert.
2	Select the type of block to insert: • Computation • Inputs • Function Blocks • Logic • Outputs
3	Left-click on the block you wish to insert.
4	Position the block in the required location in the workspace.
5	Repeat steps 1 to 5 to insert all the blocks required for the program.

Creation of Links between Blocks

At a Glance

After you have positioned the blocks in the workspace, you can link them together. To do this, you link a block's output to the input of another block. You can also loop an output back to the input of the same block.

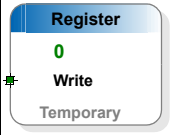
General Rules

There are some basic rules that apply when placing and connecting blocks:

- One or more connecting wires attached together form a "wire node". This is indicated in the workspace by a red dot. If wires cross without a red connection dot, it means they are not connected.
- Only one output can be attached to each wire node.
- Connections between boolean and register data are prohibited.
- Data typically flows from left to right.

Link between Blocks

The following procedure describes how to link blocks together:

Step	Action
1	Place the mouse over the first block. **Result**: One or more squares become visible on the block border, and the type of output (analog or boolean) is indicated.
2	Click the left mouse button and hold it down.
3	With the button held down, move the cursor over the input of the block you want to link to. **Result**: One or more squares become visible on the block border. If the square is green, a connection between the two blocks is possible. A red square indicates that a connection is not possible. The type of output (analog or boolean) is also indicated.  Note: Inputs and outputs have to be of the same type: a boolean output is linked to another boolean output. If the inputs or outputs are not the same, the FBD editor will display a pop-up window to indicate that origins and destinations are not of the same type.
4	Release the mouse button. **Result**: A line and a number are shown between the two linked blocks.
5	Repeat steps 1 to 4 to link all the blocks.

Link Number

There are 2 types of wires:

- The boolean wire, which will have a number beginning with B.
- The register wire, which will have a number beginning with R.

The wire number is automatically incremented in chronological order.

FBD Blocks Properties

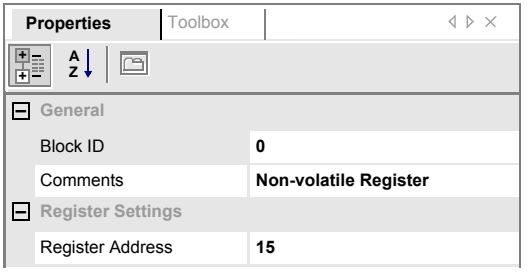
At a Glance

Each of the blocks has a properties window. To display this window, left-click on a block.

The Properties window consists of several tabs, separated in one or two categories, depending on the type of block:

- General settings, which contain the block ID and comments (common to all types of blocks.
- Specific settings ,depending on the type of block (register settings for registers, counter settings for counters, etc.).

For example, if you wish to display a non-volatile register properties, choose a non-volatile register block and left-click on it. The following window is displayed:



Comments

In the Comment zone, in the white box on the right of Comments, you can enter a comment. Select any object or any free location in the workspace to save the comment.

Settings

Most blocks have a specific settings tab. In this tab, you have to set the block's specific settings. These settings are described in detail in the help for each of the FBD blocks.

Properties Display

The properties of each block can be displayed in 2 different ways:

- by category, clicking on or
- by alphabetical order, clicking on .

FBD Resource Management

At a Glance

The LTM R controller memory is equipped with the following resources:

- Logic memory space size equal to 8192
- 300 temporary registers
- 64 non-volatile registers

Reserved Resources

When a custom logic program is developed using the structured text editor, all resources are available, whereas, when using the FBD editor, some temporary and non-volatile registers are reserved for use by the FBD compiler.

Register Allocation

The following table lists all reserved registers and their allocation. It also indicates how these registers are controlled:

Register type	Address range	Controlled by	Description
Temporary	0-69	The user	Temporary storage of bit and registers assigned by the user when creating an FBD program.
Temporary	70-299	LTM CONF	Reserved temporary registers for use by the compiler.
Non-volatile	0-31	The user	Non-volatile bits or registers assigned by the user when creating an FBD program.
Non-volatile	32-63	LTM CONF	Reserved non-volatile registers for use by the compiler.

Compiling an FBD program

At a Glance

As custom logic functions are based on the structured text language, compiling an FBD program is a 2-step process. First ,the FBD program is converted into a structured text compatible file (i.e. *.lf). This file is then compiled and downloaded into the LTM R controller (or simulator) using the structured text editor tools provided with LTM CONF programming software.

Saving an FBD Program

Before compiling the FBD program, you must save it. To save the program you created or edited, click File in the top-level menu bar and choose **Save As**.

Note: The file you saved will have a *.Gef extension.

Converting to Structured Text

To compile into structured text the program you created or edited, click Compile in the top-level menu bar and choose **To structured Text**.

A window is displayed at the bottom of the workspace with the corresponding structured text program.

Note: You can not convert a structured text program file into a FBD file.

Compiling Structured Text

Follow these steps in order to compile the structured text program just created into Pcode:

Step	Action
1	Minimize the FBD editor window and navigate in the configuration software tree (on the left of the screen in LTM CONF configuration software) to Custom Logic → Structured Text. **Result**: The program is automatically copied into the structured text editor and has a *.glf extension.
2	Compile this program by clicking **Compile**in the **Logic Functions** menu.
3	Refer to Compiling of a structured text (see *Compiling and Simulation of a Structured Text Language Program, p. 102*) program section in this manual for further information about compiling, simulation, errors check, etc.

3.4 Manipulating FBD Blocks

At a Glance

Summary

This section describes the manner in which blocks in the workspace can be manipulated, including how to select, move, duplicate or delete blocks.

What's in this Section?

This section contains the following topics:

Topic	Page
How to Select Blocks	145
How to Delete and Duplicate Objects	146

How to Select Blocks

At a Glance

When you add blocks to the workspace, you can select them to reposition them within the workspace.

How to Select One or More Blocks

The following table describes how to select one or more blocks:

If you would like to select...	Then
An isolated block	Left-click on the block.
Several contiguous blocks	Frame the blocks to be selected by defining a selection zone. **Result**: All of the selected blocks are highlighted with an orange outline.
Several blocks in different areas of the workspace	Press the Shift key, then click on the blocks to be selected while continuing to hold down the Shift key. **Result**: All of the selected blocks are highlighted with an orange outline.
All objects including wires	Select Edit -> Select All.

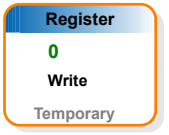
How to Delete and Duplicate Objects

At a Glance

Sometimes it may be necessary to delete a block or duplicate a block in the workspace.

How to Delete Blocks

The following table describes how to delete one or more blocks:

Step	Action
1	Select the block(s) to be deleted **Result**: The selected blocks are highlighted with an orange outline. 
2	Press the Delete or backspace key or select Edit -> Delete **Result**: The selected blocks are deleted.

How to Cut, Copy or Paste Blocks

The following table describes how to cut, copy or paste one or more blocks:

Step	Action
1	Select the block(s) to be manipulated. **Result**: The selected blocks are highlighted with an orange outline.
2	Select one of the following commands: • Edit -> Copy • Edit -> Cut • Edit -> Paste **Result**: Cut deletes the selected blocks and stores them in the clipboard. Copy duplicates the selected blocks in the clipboard and Paste duplicates the clipboard contents on the workspace. **Note:** The keyboard shortcuts Ctrl A, Ctrl C, Ctrl V and Ctrl X can also be used to copy the selected blocks, and either paste or delete them.

3.5 FBD Editor Display Options

At a Glance

Summary

The following section describes the differents FBD editor display options.

What's in this Section?

This section contains the following topics:

Topic	Page
Other Display Options	148
Workspace Appearance and Graph Options	149

Other Display Options

Summary

You can customize the following display options to suit your requirements:
- Zoom
- Links
- Inputs/Outputs

Zoom Display Options

To access zoom options, click **View** in the top-level menu bar.

3 options are offered:
- Zoom Out to see more of the program at once (shortcut: F3).
- Zoom In to focus on the program in more detail.
- Zoom To 50 %, 75 %, 100 %, 150 %, 200 % or 400 % to have a customized view of the program.

Links Display Options

To access links display options, click **Tool** in the top-level menu bar.

3 options are offered. You can:
- Renumber links, to aid in understanding the program's execution.
- Show all links, to see which blocks are linked together.
- Hide all links, to have a better overall view of the blocks.

When you click a link, its Properties window opens and enables you to customize:
- the link color,
- the text that will appear next to the link.

Inputs/Outputs Display Options

The following procedure describes how to access and change Inputs/Outputs display options:

Step	Action
1	Position the mouse over a block. **Result**: One or more squares become visible on the block border. It also indicates if the output is analog or boolean.
2	Click on this square. **Result:** The display options appear.
3	Choose if you want the label to be displayed and what text should appear.

Workspace Appearance and Graph Options

Summary	The FBD editor enables you to customize the workspace by changing its appearance and graph options.
Appearance and Graph Options	To access Appearance and Graph Options, left-click anywhere in the workspace, except on a object.

Appearance Options

The following table lists all the possible appearance customization options:

Appearance option	Description	Possible choices
Background Color	Enables you to set the background color of the workspace by clicking on the box where the color is displayed.	Choose between the colors available in the Custom, Web, and System tabs.
Background Image Path	Enables you to insert an image from your hard disk drive or any removable device and to define it as the background.	Any image you select as the background. **Note:** Only possible when the background type is set to image.
Background Type	Enables you to set the background type.	Choose between a flat color, gradient or image background.
Enable Context Menu	Shows or hides the context menu.	True or false
Enable Tooltip	Shows or hides tooltips.	True or false
Gradient Bottom	Enables you to set the color of the bottom of the gradient.	Choose between the colors available in the Custom, Web, and System tabs. **Note:** Only possible when the background type is set to gradient.
Gradient Top	Enables you to set the color of the top of the gradient.	Choose between the colors available in the Custom, Web, and System tabs. **Note:** Only possible when the background type is set to gradient.
Gradient Mode	Enables you to set the type of gradient	Choose between horizontal, vertical, forward diagonal, and backward diagonal modes. **Note:** Only possible when the background type is set to gradient.
Restrict to Canvas	Enables you to choose whether the FBD program should be kept inside the canvas.	True or false
Show Grid	Enables you to choose whether the accurate grid is visible.	True or false **Note:** This grid must not be confused with the grid line, which is accessed from the top-level **View** menu bar.
Snap	Enables you to choose whether the objects are snapped with the grid. When set to true, if you move objects, they will move along the grid step.	True or false

Graph Options The following table lists all the possible graph customization options:

Graph option	Description	Possible choices
Allow Add Connection	Enables you to choose whether connections can be added to the workspace.	True or false
Allow Add Shape	Enables you to choose whether blocks can be added to the workspace.	True or false
Allow Delete Shape	Enables you to choose whether blocks can be deleted.	True or false
Allow Move Shape	Enables you to choose whether blocks can be moved in the workspace.	True or false
Locked	Enables you to choose whether the FBD program can be edited.	True or false

Display Grid You may wish to display the grid lines. In order to do so, select View -> Display Grid.

Connection to the LTM R Controller

4

At a Glance

Overview

This chapter describes how to connect the HMI device running custom logic editor to the LTM R controller. It details how to physically connect the device to the controller, including what connection accessories can be used, as well as describing how to transfer logic files between the LTM R controller and the custom logic editor.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Hardware Connection	154
Initialization and Connection	156
Transferring Logic Files between the LTM R Controller and Custom Logic Editor	158
Custom Logic Program Transfer and Execution	162

Hardware Connection

Overview

This section describes how to physically connect the LTM R controller to a PC running Powersuite™ or LTM CONF.

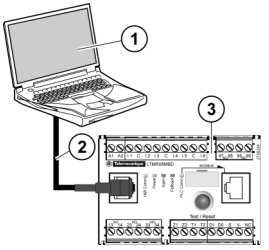
The PC requires its own power source and must be connected to the RJ45 port on the LTM R controller or the HMI interface port (RJ45) on the expansion module when attached to the LTM R controller.

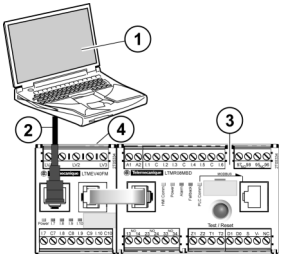
Configurations

The PC can be connected in a 1-to-1 configuration to a single LTM R controller, or in a 1-to-many configuration to multiple controllers.

Connecting to a PC running LTM CONF Software in 1-to-1 Mode

The diagrams below show a 1-to-1 connection from a PC running LTM CONF to the LTM R controller, with and without the expansion module:

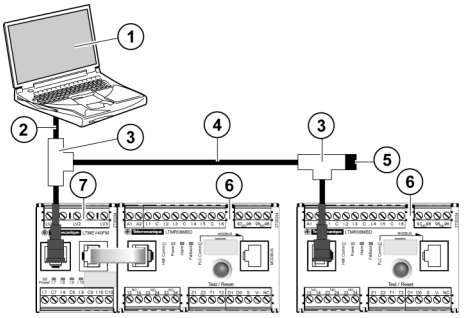




1 PC running LTM CONF software
2 Power cable VW3 A8 106
3 LTM R controller
4 Expansion module

Connecting to a PC running LTM CONF Software in 1-to-Many Mode

The diagram below shows a 1-to-many connection from a PC running LTM CONF software to up to 8 controllers (with or without the expansion module):



1 PC running LTM CONF software
2 Power cable VW3 A8 106
3 T-junction boxes VW3 A8 306 TF••
4 Communication cable VW3 A83 06R••
5 line terminators VW3 A8 306 R
6 LTM R controller
7 Expansion module

Connection Accessories

The following table lists connection accessories:

Designation	Description	Reference
T-junction boxes	With 0.3 m (1 ft) integrated cable	VW3 A8 306 TF03
	With 1 m (3.2 ft) integrated cable	VW3 A8 306 TF10
Line terminators for RJ45 connector	R = 150 Ω	VW3 A8 306 R
Power cable (PC only)	Length = 1 m (3.2 ft) RS-232 to RS-485 converter	VW3A8106
Communication cables	Length = 0.3 m (1 ft)	VW3 A8 306 R03
	Length = 1 m (3.2 ft)	VW3 A8 306 R10

Initialization and Connection

Initialization

When you connect the LTM R controller to the PC, the controller automatically initializes. This initialization process enables the controller and the PC to exchange identification information.

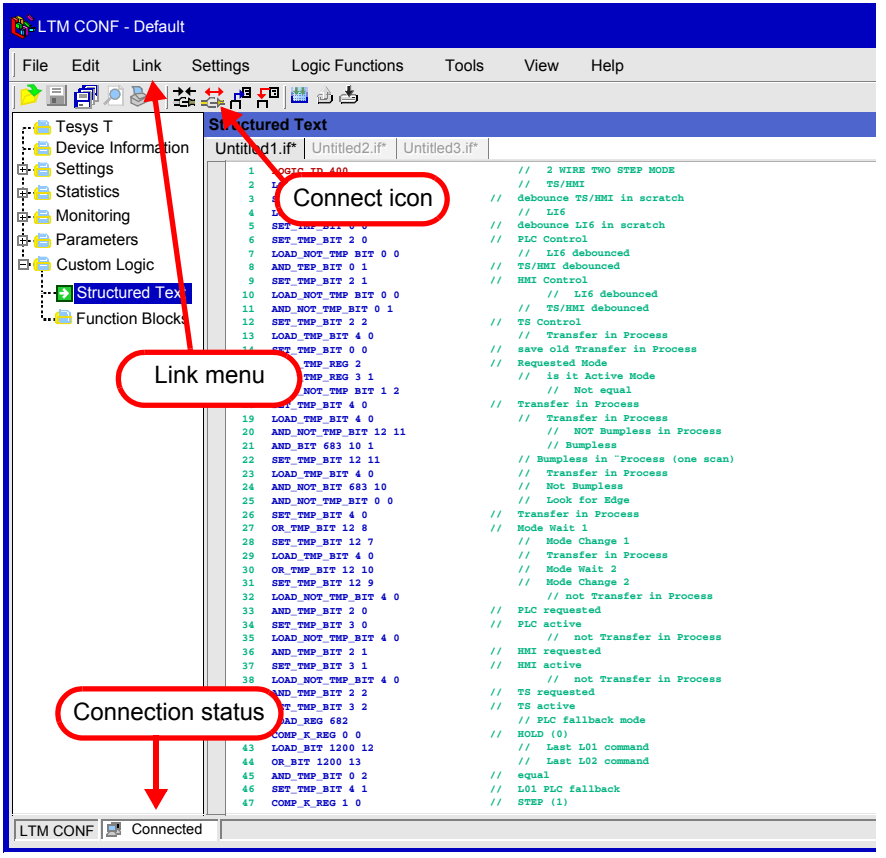
During this process, the custom logic editor indicates "Wait" until initialization is complete.

Connection

After initialization, the LTM R controller should automatically connect to the PC.

To verify that the controller is connected, check the task bar in the custom logic editor.

If the task bar reads **Not connected**, then select Connect on the Link menu or click the Connect icon.



A progress bar briefly appears as your PC connects to the controller, and the word **Connected** appears in the task bar when the connection process successfully completes.

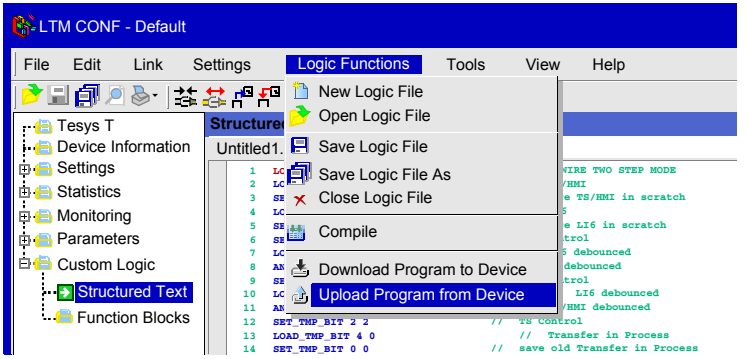
When the LTM R controller is connected, you can

- upload custom logic files from the controller to LTM CONF software for editing
- download edited custom logic files from LTM CONF software to the controller

Transferring Logic Files between the LTM R Controller and Custom Logic Editor

File Transfer - LTM R Controller to PC

To transfer logic files from the LTM R controller to the custom logic editor:

Step	Action
1	Ensure that the LTM R controller is connected to the PC. (See *p. 157*.)
2	Select **Logic Functions** → **Upload Program from Device** or click the icon to transfer the logic file from the LTM R controller to the custom logic editor. 
3	When the logic file has been transferred, you can use custom logic editor to change configuration settings.
4	After your logic file edits are complete, save your work to a file. Select the **Save** command in either the icon bar or the **File** menu, navigate to the desired location and click **Save**.

Saving Files

Save a copy of any logic file you intend to transfer to the LTM R controller. A saved copy provides both a record of these settings, and a backup that can be used to re-transfer configuration settings if the initial transfer fails.

Use the:

- **Save** command to save your changes to the open configuration file
- **Save As** command to save a copy of the displayed configuration to a separate file.

Note: If you opened the file containing the factory default configuration settings, you cannot make and save changes to this file. Instead, you must use the **Save As** command to save your changes under another file name.

By default, the configuration software stores saved files in a folder named "Configurations". This folder is located on your hard drive in the same place the configuration software was installed.

To designate a different default file storage folder:

Step	Action
1	In the Settings menu, select **Preferences**. The Preferences dialog opens.
2	In the Preferences dialog, open the Configuration tab.
3	In the Configuration tab type in the folder name and path for saving configuration files.
4	Click **OK** to close the Preferences dialog and save your changes.

File Transfer - PC to LTM R Controller

After you have edited and compiled your logic file, you can transfer the file to the LTM R controller. Before the configuration software will make this transfer, the following conditions must be met:

- At least one setting in the logic file must be different from the corresponding setting in the controller - i.e., the software only overwrites settings with different values.
- Current must not be detected - that is, online current must be less than 10% of FLC.

If these conditions are not met, the file cannot be transferred to the controller.

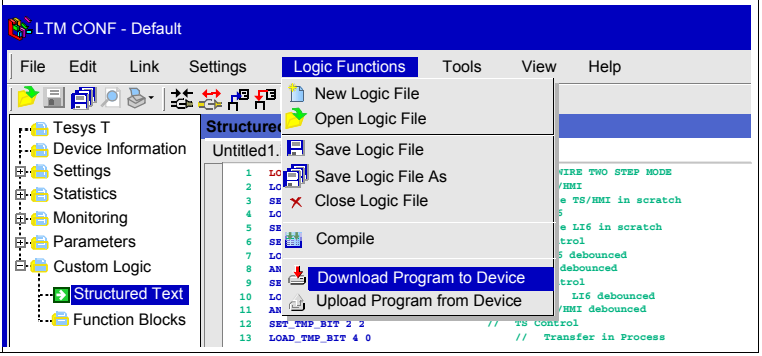
Note: When you transfer a configuration file from the PC to the LTM R controller, the software checks to confirm that the LTM R controller and the configuration file both use the same:

- current range, and
- network protocol

If there is a mismatch, the software asks if you wish to proceed. If you elect to proceed, the software transfers all matching parameters, excluding parameters that fail a range check. When the transfer is complete, the software displays the names and addresses of parameters that failed the range check and were not transferred.

File Transfer Procedure - PC to LTM R Controller

To transfer a logic file from the custom logic editor to the LTM R controller:

Step	Action
1	Ensure that the LTM R controller is connected to the PC. (See *p. 157*.)
2	Ensure that the file to be transferred is in the Main window. To open a file, select the **Open Configuration** command in either the icon bar or the File menu. Then navigate to the desired location and click **Open**.
3	Select **Logic Functions** → **Download Program to Device** or click the icon to transfer the logic file from custom logic editor to the controller. 

Custom Logic Program Transfer and Execution

Overview

Custom logic programs may be uploaded to or downloaded from the LTM R controller via LTM CONF configuration software. Only one custom logic program may be loaded into the LTM R controller at a time.

Transfer Validity Check

During the upload or download of a custom logic program, outputs are turned off and logic execution is stopped.

A specific mechanism is used to upload or download a custom logic file. This mechanism uses a size register, checksum and custom logic ID (see *Characteristics of the Custom Logic Program, p. 24*) code to ensure that an incomplete or corrupt logic function can be detected. LTM CONF configuration software will not allow a logic file to be uploaded with a bad checksum, however, interrupting the connection during the upload will be detected by the checksum mechanism.

Custom Logic Program Selection

Once a custom logic file is uploaded to the LTM R controller, that program may be selected by choosing "Custom" from the motor controller mode selection menu or by writing its logic ID (see *p. 17*) code to register 540.

Custom Logic Program Replacement

In the situation where a custom logic program is replaced by another one with a different logic ID code and the installed custom program is selected, when the new program is uploaded, the value in register 540 is automatically changed to the new logic ID code. In cases, when a standard motor controller mode is currently active (i.e. Logic ID = 2 through 11) the value in register 540 does not change.

Invalid Program

If the custom logic program that is stored in memory has a bad checksum, an invalid size or invalid logic ID, or if there is no program stored in memory, it is impossible to select "Custom" from the motor controller mode selection menu. Writing a logic ID value to register 540 that does not match one of the pre-defined operating modes or the logic ID of the valid, checksummed custom logic program in memory is blocked by the LTM R controller.

Corrupted Program

If the custom logic program in memory is already selected and becomes corrupted (either by loading a corrupt function over it or by data loss in the memory) then the LTM R controller issues a minor internal fault as soon as the corruption is detected.

Appendices

Custom Logic Pre-Defined Operating Modes Programs

Overview

The LTM R supports 10 pre-defined operating modes. The following chapter contains shows their program in structured text language.

What's in this Appendix?

The appendix contains the following chapters:

Chapter	Chapter Name	Page
A	LTM R Controller Programming Approach	165
B	Pre-Defined Operating Modes Structured Text Programs	179

LTM R Controller Programming Approach

A

At a Glance

Overview

These appendices include the Pcode files used for the 10 pre-defined operating modes. To help introduce the general programming strategy for each of the 10 pre-defined operating mode, this chapter includes a summary of the strategy used for one program - the 3-wire independent operating mode which is the most commonly used. While the other programs may have more or less program steps, a general understanding of this program will help you in your attempt to analyze any of the 10 programs for the pre-defined operating modes.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
LTM R Controller Programming Strategy for the 3-Wire Independent Operating Mode	166
3-Wire Independent Mode Programming Example	168
3-Wire Independent Operating Mode Temporary Registers Allocation	170
3-Wire Independent Operating Mode Program Sections	173
3-Wire Independent Operating Mode Allocation Tables	175

LTM R Controller Programming Strategy for the 3-Wire Independent Operating Mode

Overview

The 3-wire independent operating mode program is made up of several parts. Usually, a program is divided as follows:

1. Temporary registers allocation.
2. Program in itself (logic commands and arguments) divided by sections and explained by comments.

Comments

The Comments are indicated by "//" marked in the program.

Initial Comments

The initial comments describe the temporary address locations (see *3-Wire Independent Operating Mode Temporary Registers Allocation, p. 170*) needed to perform specific functions of the program.

3-Wire Independent Mode Structure

While there are several hundred logic commands and arguments, there are essentially 9 sections (see *3-Wire Independent Operating Mode Program Sections, p. 173*) for the 3-wire independent operating mode program. Each of the 9 sections includes a summary comment that labels each section of the program.

3-Wire Independent Mode Sections

The general strategy for each section is described further in this chapter (see *3-Wire Independent Operating Mode Program Sections, p. 173*), followed by an address table (see *3-Wire Independent Operating Mode Allocation Tables, p. 175*) section which defines the inputs and outputs in each segment of the Pcode (program).

Purpose of the Programming Strategy

This information is intended to help you to understand the program's objectives well enough to quickly locate the section of the code and the variables that you wish to modify.

Example: In the 3-wire independent mode, make the LTM R Terminal Strip input 3 generate an external fault condition. "External fault" bit will activate fault bit 455.2, disabling LO4 on the next scan.

General Rules

There are several general programming techniques used in the operating modes.

However, some rules are common to all of them:

- The default state for all custom logic memory addresses is zero.
- If the logic does not direct the state of the bit as True, the status of the bit is considered to be False and set to zero.
- The LTM R controller scans the logic program from top to bottom.

New Commands

Some programming functions are only active when a new command is detected. To detect a new command, the status of the variable as a result of the previous scan, is saved to history. The status of the variable in the current scan is compared to the status of the last scan.

Debounced Bits

Several bits are "debounced" into a temporary scratch register to insure that the state of the inputs are latched and do not change during the period in which the LTM R controller microprocessor executes the sequential logic.

Note: Address locations used as scratch address locations may be used in one section of the program for one purpose and then used in later sections of the program for a second purpose.

3-Wire Independent Mode Programming Example

Overview

The purpose of this example is to familiarize you, by using the wiring diagrams and description of the pre-defined operating modes in the LTM R user's manuals, with the relationship between the LTM R controller physical and logical inputs and outputs used to control a Direct across the line (FVNR) motor starter.

You will find that use of the recommended power and control wiring diagrams is essential to implementing the logic contained in the predefined operating mode Pcode.

What we want to do in this example, is to make the LTM R Terminal Strip input 3 generate an external fault condition. "External fault" bit will then activate fault bit 455.2, disabling LO4 on the next scan.

Wiring Requirements

The control circuit requires LO4 in series with LI4 - to cause a STOP command which disables the firmware latch on LO1, which is wired in series with the contactor coil.

Program Scan

The following table describes the actions performed by the LTM R control program:

Stage	In operation the LTM R controller
1	scans the address inputs,
2	executes program saved to memory logic addresses,
3	sets the status of output addresses and directs the physical and logic outputs of the LTMR according to the state of the logic outputs in register 1200.

Program Sections

Use the section (see *3-Wire Independent Operating Mode Program Sections, p. 173*) titles to familiarize yourself with the general strategy for the 3-wire Pcode control program. For more complex modifications you may wish to become more familiar with some of the common programming techniques as noted below.

Using the Program Sections

To make the LTM R Terminal Strip input 3 generate an external fault condition:

Step	Action
1	Locate the main section of the Pcode which manages the behavior of the LTM R controller outputs (see *Program Sections, p. 173*). **Result**: You will find that the section of the program titled "Set Logic Outputs in the LTM R" includes the logic output to initiate an LTM R external fault at address 1200.8.
2	Locate the address representing the status of physical input 3, using the variables map in the LTMR user manual. **Result**: You will find that the address bit is 457.2.
3	Open the PCode for the 3-wire independent operating mode (see *p. 215*) into the structured text editor (see *Introducing the Structured Text Editor, p. 31*).
4	Add the following programming steps to the "Set LTRM Outputs" section. Load_bit 457.1 //Load the Status of logic input 457.1 into the accumulator. Set_bit 1200.8 //Set the bit to cause an LTM R external fault condition.
5	Compile the program and save the file under a new name.
6	Test the program in the logic simulator (see *p. 112*) to verify the desired behavior of the inputs and outputs.
7	Download the program from PC into the LTM R controller and wire the LTM R controller for use as desired.

3-Wire Independent Operating Mode Temporary Registers Allocation

Overview

Usually, at the beginning of the program, the initial comments describe the temporary address locations needed to perform specific functions of the program.

Temporary Registers Allocation

The following comments describe the temporary registers allocation for the 3-wire independent operating mode:

```
LOGIC_ID 5  // 3-WIRE INDEPENDENT MODE
// Temp register allocation
// Temp 0 and Temp 1 as scratch
// Temp 2 as Requested Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
// Temp 3 as Active Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
// Temp  4 as state bits group 1
//       0=Control Transfer in process
//       1=LO1 PLC fallback value
//       2=L02 PLC fallback value
//       3=L01 HMI fallback value
//       4=L02 HMI fallback value
//       5=Global Stop
//       6=Stop1
//       7=Stop2
//       8=Run1
//       9=Run2
//       10 = Unused
//       11 = Unused
//       12 = Unused
//       13 = Swapping in process (always 0 in this program).
//
// Temp 5 as state bits group 2 - Unused in this program.
//
// Temp 12 as INPUT History,
//       1=PLC Run 1
//       2=PLC Run 2
//       3=HMI Run 1
//       4=HMI Run 2
//       5=TS Run 1
//       6=TS Run 2
//       7=Bump Mode Change for Output 1
//       8= Unused
//       9=Bump Mode Change for Output 2
//       10=Unused
//       11=Bumpless mode in Process
//       12=Power-up Done
//
// Temp 50+ as general status registers
// Temp 50  as ONSET status transition time value
// Temp 51  as ONSET status Low to High timer
// Temp 52  as ONSET status High to Low timer
// Temp 53  Latch
// Temp 54  as Mask off Reg 704 Run1,Run2 in bumped transfers
//
```

Temporary Register 4 bit 13

The bit 13 of the temporary register 4 (swapping in process) is always to 0 in the 3-wire independent operating mode. This bit is used in the 2-wire 2-speed (see *p. 229*), 3-wire 2-speed (see *p. 300*), 2-wire reverser (see *p. 229*) and 3-wire reverser (see *p. 245*) pre-defined operating modes. In these programs, the PCode manages whether or not a STOP is required to change or SWAP a command from LO1 to LO2.

Temporary Register 50 to 54

Temporary address locations 50, 51, 52, 53 are used in the 2-speed, reverser or 2-step pre-defined operating modes. In these programs, the PCode must detect that the appropriate transition timers have elapsed before allowing a SWAP a command from LO1 to LO2. These comments are shown in the 3-wire independent operating mode PCode to help you to locate where in the other programs you should use these temporary addresses in the event you would like to use the Pcode from those programs in a similar manner within the 3-wire independent operating mode.

3-Wire Independent Operating Mode Program Sections

Overview

The 3-wire independent operating mode is made up of 9 sections.

Program Sections

Each section of the program enables the LTM R controller to perform an action or to define a specific function.

The function of each section is described below:

```
//Manage Requested and Active Control Source
   //Detect Local/Remote settings and save Requested Control Source in Temp Reg 2
   //Detect if 'Request' NOT equal 'Active' source. Save 'Transfer in Process' Temp 4.0
   //Save Active Control Source in Temp Reg 3

//Manage Bump/Bumpless transfers
   //Save active transfers as bumpless "transfer in process" in Temp Reg 12.11
   //Detect Bump config, Save "bump transfer in process" in Temp Reg 12.7, Temp Reg 12.9

//Manage Communications Loss-Fallback Values
   //Save PLC Fallback Settings for LO1 and LO2 in Temp Reg 4.1 and 4.2
   //Save HMI Fallback Settings for LO1 and LO2 in Temp Reg 4.3 and 4.4

//Latch HMI Keypad input
   //Latch HMI Aux1, Aux2 and Stop Inputs into in Temp Reg 13.12, 13.13 and 13.14

//Generate Stop Commands
   //Manage Global Stop
       //Detect Global Stop Inputs, SET Global Stop in Temp Reg 4.5
   //Latch comm loss values for use in Stop and Run Commands
       //Detect PLC and HMI Comm Loss. Save status to Temp Reg bit 0.0 and 0.1
   //Generate Stop1 Commands
       //Detect Stop1 input status. Save Stop1 command to Temp Reg 4.6
   //Generate Stop2 Commands
       //Detect Stop 2 input status. Save Stop2 Command to Temp Reg 4.7
```

Program Sections (cont'd)

```
//Generate Run 1 And Run 2 Commands

   //Generate Run 1 - PLC/HMI/TS Mode
       //Save previous Run1 history in Temp Reg 12.0
      //Save new Run2 history in Temp Reg 12.1 (PLC), 12.3 (HMI) or 12.5 (TS)
       //If new Run1 command, Save partial Run 1 to Temp Reg 4.8
       //3wire latch Run1
       //Detect No Run1 inhibits. Save final Run 1 to Temp Reg 4.8

   //Generate Run 2 PLC/HMI/TS Mode
       //Save previous Run2 history in Temp Reg 12.0
      //Save new Run2 history in Temp Reg 12.2 (PLC), 12.4 (HMI) or 12.6 (TS)
       //If new Run2 command, Save partial Run2 to Temp Reg 4.9
       //3wire latch Run 2
       //Detect no Run2 inhibits. Save final Run 2 to Temp Reg 4.9

// Set Outputs to IMPR
   //Process Output 1
       //If Run1 (Temp 4.8) active and NO Stop 1 command (Temp 4.6)
      //SET Output 1(1200.12) Aux1 LED (1200.9) Motor Run (1200.0), Motor Stop (1200.1)
   //Process Output 2
       //IF Run2 (Temp 4.9) active and NO Stop 2 Command (4.7)
       //SET Output 2 (1200.13) and Aux 2 LED ON (1200.10)
   //Process Outputs 3(Warn) and Ouput4 (Fault)
       //If IMPR Warn status active, Set LO3 ON (1200.14)
       //If IMPR Fault status active, Set LO4 OFF (1200.15)
   //Process Logic Reset
       //If Local Reset command active Set Logic Reset  (1200.2)
   //Process Remote Status
       //If PLC active Set Remote Control Source active (1200.6)
   //Process STOP Logic
      //If Stop 1 (Temp 4.6) OR Stop 2 (Temp 4.7) active Set Stop LED ON (1200.11)

// Manage Power-UP Done
   //If NO global Stops, OR Control Source Active, Set Power-up Done Temp Register 12.12

// Clear PLC Control on Control Transfer
   //If Control Source Transfer active AND No Bumpless Cfg, Mask off Reg 704 Run1,Run2
```

3-Wire Independent Operating Mode Allocation Tables

Overview

Each allocation table defines the inputs and outputs values in each section of the 3-wire independent operating mode.

Allocations Tables

The allocation tables for each section of the 3-wire independent operating mode are described below:

//Manage Requested and Active Control Source

Inputs		Outputs	
Address	**Description**	**Address**	**Description**
457.5	Remote	T 2.0	Remote Request
683.8	TS/HMI	T 2.1	HMI Request
		T 2.2	TS Request
		T 3.0	Remote Active
		T 3.1	HMI Active
		T 3.2	TS Active

Manage Bump and Bumpless

Inputs		Outputs	
Address	**Description**	**Address**	**Description**
683.10	Bumpless	T 4.0	Transfer Active
		T 12.11	Bumpless in process
		T 12.7	Bump LO1 in process
		T 12.9	Bump LO2 in process

//Manage Communications Loss-Fallback Values

Inputs		Outputs	
Address	**Description**	**Address**	**Description**
R682	PLC Fallback Settings	T 4.1	PLC FB LO1
		T 4.3	PLC FB LO2
R645	HMI FB Settings	T 4.2	HMI FB LO1
		T 4.4	HMI FB LO2

//Latch HMI Keypad input

Inputs		Outputs	
Address	**Description**	**Address**	**Description**
1020.14	HMI Run 1	T 13.12	HMI Run 1
1020.12	HMI Run 2	T 13.13	HMI Run 2
1020.13	HMI STOP	T 13.14	HMI STOP

////Manage Global Stop

Inputs		Outputs	
Address	**Description**	**Address**	**Description**
T 13.14	HMI STOP	T 4.5	Global STOP
456.5	Load Shed		

////Generate Stop1 Commands

Inputs		Outputs	
Address	**Description**	**Address**	**Description**
T 4.5	Global Stop	T 4.6	STOP 1
453.1	Diag 1		
453.2	Diag 2		
456.4	RC Timer		
456.8	PLC Loss		
457.6	HMI Loss		
T 4.1	PLC FB STOP		
T 4.3	HMI FB STOP		
1200.12	Logic Stop		

////Generate Stop2 Commands

Inputs		Outputs	
Address	**Description**	**Address**	**Description**
T 4.5	Global Stop	T 4.7	STOP 1
456.8	PLC Loss		
457.6	HMI Loss		
T 4.2	PLC FB STOP		
T 4.4	HMI FB STOP		

////Generate Run1

Inputs		Outputs	
Address	**Description**	**Address**	**Description**
T 12.12	Power Up	T 4.8	Run1
T 3.0, T 3.1, T 3.2	Active Source		
704.0	PLC Run 1		
T 13.12	HMI Run 1		
457.0	TS Run 1		
T 12.0	Edge Detected1		
T 4.12	No Lock Outs		
456.8	PLC Loss		
457.6	HMI Loss		
T 4.8	PLC FB Run LO1		
T 4.10	HMI FB Run LO1		
T 12.7	Bump LO1 in process		
12. 11	Bumpless in process		

////Generate Run2

Inputs		Outputs	
Address	**Description**	**Address**	**Description**
T 12.12	Power Up	T 4.9	Run2
T 3.0, T 3.1, T 3.2	Active Source		
704.1	PLC Run 2		
T 13.13	HMI Run 2		
457.1	TS Run 2		
T 12.0	Edge Detected1		
T 4.12	No Lock Outs		
456.8	PLC Loss		
457.6	HMI Loss		
T 4.9	PLC FB Run LO2		
T 4.11	HMI FB Run LO2		
T 12.9	Bump LO2 in process		
12. 11	Bumpless in process		

///Set Logic Outputs to IMPR

Inputs		Outputs	
Address	**Description**	**Address**	**Description**
T 4.8	Run 1	1200.12	LO1
T 4.9	Run 2	1200.13	LO2
T 4.6	Stop 1	1200.14	LO3
T 4.7	Stop 2	1200.15	LO4
455.2	Warn	1200.0	Run
455.3	Fault	1200.1	Stop
457.4	LI5 Reset	1200.2	Reset
T 3.0	PLC Active	1200.5	Direction
		1200.6	Remote
		1200.7	FLA Set
		1200.8	Ext Fault
		1200.9	Aux1 LED
		1200.10	Aux2 LED
		1200.11	Stop LED

///Manage Power-UP Done

Inputs		Outputs	
Address	**Description**	**Address**	**Description**
T 4.5	Global Stop	T 12.12	Power-up
T 4.0	Transfer Active		

// Clear PLC Control on Control Transfer

Inputs		Outputs	
Address	**Description**	**Address**	**Description**
T 4.0	Transfer Active	704	Commands
683.10	Bumpless	T 54	ON Set
704	Command Reg		

Note: Single bit designations are represented by an address and decimal. For example 457.4 is address 457, bit 4. Sixteen bit word designations are represented by the designation "R" followed by the address number without a decimal. For example R682 is the 16 bit word at address 682.

Pre-Defined Operating Modes Structured Text Programs

B

At a Glance

Overview

The LTM R controller supports 10 pre-defined operating modes (See the section Motor Control Functions in the LTM R Motor Management Controller Reference Manual).

The following chapter shows the structured text program for each pre-defined operating mode.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Structured Text Program for 2-Wire Overload Mode	180
Structured Text Program for 3-Wire Overload Mode	191
Structured Text Program for 2-Wire Independent Mode	202
Structured Text Program for 3-Wire Independent Mode	215
Structured Text Program for 2-Wire Reverser Mode	229
Structured Text Program for 3-Wire Reverser Mode	245
Structured Text Program for 2-Wire 2-Step Mode	261
Structured Text Program for 3-Wire 2-Step Mode	273
Structured Text Program for 2-Wire 2-Speed Mode	285
Structured Text Program for 3-Wire 2-Speed Mode	300

Structured Text Program for 2-Wire Overload Mode

Overview

The structured text program for the 2-wire overload mode is defined below:

Structured Text Program

```
LOGIC_ID 2  // 2-WIRE OVERLOAD MODE
// Temp register allocation
// Temp 0 and Temp 1 as scratch
// Temp 2 as Requested Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
// Temp 3 as Active Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
// Temp  4 as state bits group 1
//       0=Control Transfer in process
//       1=LO1 PLC fallback value
//       2=L02 PLC fallback value
//       3=L01 HMI fallback value
//       4=L02 HMI fallback value
//       5=Global Stop
//       6=Stop1
//       7=Stop2
//       8=Run1
//       9=Run2
// Temp 5 as state bits group 2
//
// Temp 12 as INPUT History
//       1=PLC Run 1
//       2=PLC Run 2
//       3=HMI Run 1
//       4=HMI Run 2
//       5=TS Run 1
//       6=TS Run 2
//       7=Mode Change 1
//       8=
```

Structured Text Program (cont'd)

```
//       9=Mode Change 2
//       10=
//       11=Bumpless in Process
//       12=Power-up Done
//
// Temp 50+ as general status registers
// Temp 50  as ONSET status transition time value
// Temp 51  as ONSET status Low to High timer
// Temp 52  as ONSET status High to Low timer
// Temp 53  Latch
// Temp 54  as ONSET status  704 Run1-Run2
//
//Save Requested Control.in Temp 2
//
LOAD_BIT 683.8          //TS/HMI
SET_TMP_BIT 0.1         //Debounce TS/HMI in scratch
LOAD_BIT 457.5          //LI6
SET_TMP_BIT 0.0         //Debounce LI6 in scratch
SET_TMP_BIT 2.0         //PLC Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_TMP_BIT 0.1         //TS/HMI debounced
SET_TMP_BIT 2.1         //HMI Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_NOT_TMP_BIT 0.1     //TS/HMI debounced
SET_TMP_BIT 2.2         //TS Control
//
//Look for control transfer
//
LOAD_TMP_BIT 4.0        // Transfer in Process
SET_TMP_BIT 0.0         //save old Transfer in Process
LOAD_TMP_REG 2          //Requested Mode
COMP_TMP_REG 3, 1       //is it Active Mode
LOAD_NOT_TMP_BIT 1.2    //Not equal
SET_TMP_BIT 4.0         //Transfer in Process
//
```

Structured Text Program (cont'd)

```
//Manage Bump/Bumpless
//
LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_TMP_BIT 12.11   //NOT Bumpless in Process
SET_TMP_BIT 12.11       //Bumpless in Process (one scan)

LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_BIT 683.10      //Not bumpless
AND_NOT_TMP_BIT 0.0     //Look for Edge
SET_TMP_BIT 4.0         //Transfer in Process
SET_TMP_BIT 12.7        //Mode Change 1
SET_TMP_BIT 12.9        //Mode Change 2
//
// Save Active Control Mode in Temp Reg 3
//
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.0         //PLC requested
SET_TMP_BIT 3.0         //PLC active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.1         //HMI requested
SET_TMP_BIT 3.1         //HMI Active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.2         //TS requested
SET_TMP_BIT 3.2         //TS active
//
```

Structured Text Program (cont'd)

```
// Generate PLC Fallback Values
//
LOAD_REG 682              //PLC fallback mode
COMP_K_REG 0, 0           //---HOLD(0)---
LOAD_TMP_BIT 0 2          //equal
AND_BIT 1200.12           //last LO1 command
SET_TMP_BIT 4.1           //LO1 PLC fallback
LOAD_TMP_BIT 0 2          //equal
AND_BIT 1200.13           //last LO2 command
SET_TMP_BIT 4.2           //LO2 PLC fallback
                          //---STEP(1)--- no action needed
                          //---OFF(2)---- no action needed
COMP_K_REG 3, 0           //---ON(3)-----
LOAD_K_BIT 1              //fallback to ON
AND_TMP_BIT 0 2           //equal
OR_TMP_BIT 4.1            //logical or with previous value
SET_TMP_BIT 4.1           //LO1 PLC fallback
LOAD_K_BIT 1              //fallback to ON
AND_TMP_BIT 0 2           //equal
OR_TMP_BIT 4.2            //logical or with previous value
SET_TMP_BIT 4.2           //LO2 PLC fallback
COMP_K_REG 4, 0           //---ON OFF(4)----
LOAD_K_BIT 1              //fallback to ON
AND_TMP_BIT 0 2           //equal
OR_TMP_BIT 4.1            //logical or with previous value
SET_TMP_BIT 4.1           //LO1 PLC fallback
COMP_K_REG 5, 0           //---OFF ON(5)----
LOAD_K_BIT 1              //fallback to ON
AND_TMP_BIT 0 2           //equal
OR_TMP_BIT 4.2            //logical or with previous value
SET_TMP_BIT 4.2           //LO2 PLC fallback

//
```

Structured Text Program (cont'd)

```
// Generate HMI Fallback Values
//
LOAD_REG 645            //HMI fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.12         //last LO1 command
SET_TMP_BIT 4.3         //LO1 HMI fallback
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.13         //last LO2 command
SET_TMP_BIT 4.4         //LO2 HMI fallback
                        //---STEP(1)--- no action needed
                        //---OFF(2)---- no action needed
COMP_K_REG 3, 0         //---ON(3)-----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.3          //logical or with previous value
SET_TMP_BIT 4.3         //LO1 HMI fallback
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.4          //logical or with previous value
SET_TMP_BIT 4.4         //LO2 HMI fallback
COMP_K_REG 4, 0         //---ON OFF(4)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.3          //logical or with previous value
SET_TMP_BIT 4.3         //LO1 HMI fallback
COMP_K_REG 5, 0         //---OFF ON(5)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.4          //logical or with previous value
SET_TMP_BIT 4.4         //LO2 HMI fallback

//
```

Structured Text Program (cont'd)

```
// Latch HMI Keypad info
//
LOAD_BIT 1020.12        //Aux 1
SET_TMP_BIT 13.12
LOAD_BIT 1020.13        //Aux 2
SET_TMP_BIT 13.13
LOAD_BIT 1020.14        //Stop
SET_TMP_BIT 13.14


//
// Generate Global Stop in Temp Reg 4.5
//
LOAD_TMP_BIT 3.1        //HMI Active
AND_BIT 455.2           //IMPR Fault status
OR_TMP_BIT 13.14        //HMI Stop Key
OR_BIT 456.5            //Load Shed
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_TMP_BIT 3.0    //NOT PLC active
AND_NOT_TMP_BIT 3.1     //NOT HMI active
AND_NOT_TMP_BIT 3.2     //NOT TS active
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save final Global Stop


//
//Latch comm loss values in scratch 0
//
LOAD_BIT 456.8          //PLC Comm Loss
SET_TMP_BIT 0.0         //save in scratch bit 0
LOAD_BIT 456.7          //HMI Comm Loss
SET_TMP_BIT 0.1         //save in scratch bit 1

//
```

Structured Text Program (cont'd)

```
// Generate Stop1 and Stop2 Commands
//
                          //Generate Stop1
LOAD_TMP_BIT 4.5          //Global Stop
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_TMP_BIT 0.0          //PLC Comm Loss from scratch
AND_TMP_BIT 3.0           //PLC active
AND_NOT_TMP_BIT 4.1       //NOT LO1 PLC fallback value
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_TMP_BIT 0.1          //HMI Comm Loss from scratch
AND_TMP_BIT 3.1           //HMI active
AND_NOT_TMP_BIT 4.3       //NOT LO1 HMI fallback value
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1

LOAD_TMP_BIT 3.0          //PLC active
AND_NOT_BIT 704.0         //NOT PLC Run1
AND_TMP_BIT 4.8           //Run 1
AND_NOT_TMP_BIT 0.0       //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1

LOAD_TMP_BIT 3.1          //HMI active
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1

LOAD_TMP_BIT 3.2          //TS active
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save final Stop1
```

Structured Text Program (cont'd)

```
                                //Generate Stop2
LOAD_TMP_BIT 4.5                //Global Stop
SET_TMP_BIT 4.7                 //save partial Stop7
LOAD_TMP_BIT 0.0                //PLC Comm Loss from scratch
AND_TMP_BIT 3.0                 //PLC active
AND_NOT_TMP_BIT 4.2             //NOT LO2 PLC fallback value
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2
LOAD_TMP_BIT 0.1                //HMI Comm Loss from scratch
AND_TMP_BIT 3.1                 //HMI active
AND_NOT_TMP_BIT 4.4             //NOT LO1 HMI fallback value
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2

LOAD_TMP_BIT 3.0                //PLC active
AND_NOT_BIT 704.1               // NOT PLC Run2
AND_TMP_BIT 4.9                 //Run 2
AND_NOT_TMP_BIT 0.0             //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2

LOAD_TMP_BIT 3.1                //HMI active
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2

LOAD_TMP_BIT 3.2                //TS active
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save final Stop2


//
```

Structured Text Program (cont'd)

```
// Generate Run1 and Run2 Commands
//
                          //Generate Run 1
                          //PLC mode
LOAD_TMP_BIT 12.1         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 704.0            //PLC Network Run1
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.1          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.0           //PLC active
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

                          //PLC Fallback
LOAD_TMP_BIT 4.1          //PLC fallback value
AND_TMP_BIT 3.0           //PLC active
AND_TMP_BIT 0.0           //PLC Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run 1

                          //HMI Fallback NA

                          //3wire latch
AND_NOT_TMP_BIT 4.13      //NOT Swapping
AND_NOT_TMP_BIT 12.7      //NOT Mode Change 1
SET_TMP_BIT 4.8           //save final Run 1
```

Structured Text Program (cont'd)

```
                                //Generate Run 2
                                //PLC mode
LOAD_TMP_BIT 12.2               //Input history
AND_NOT_TMP_BIT 12.11           //NOT Bumpless in Process
SET_TMP_BIT 12.0                //Save previous history
LOAD_BIT 704.1                  //PLC Network Run2
AND_TMP_BIT 12.12               //Power-up Done
SET_TMP_BIT 12.2                //Save new history
AND_NOT_TMP_BIT 12.0            //NOT previous history
AND_TMP_BIT 3.0                 //PLC active
OR_TMP_BIT 4.9                  //Include previous result
SET_TMP_BIT 4.9                 //save partial Run2

                                //PLC Fallback
LOAD_TMP_BIT 4.2                //PLC fallback value
AND_TMP_BIT 3.0                 //PLC active
AND_TMP_BIT 0.0                 //PLC Comm Loss from scratch
OR_TMP_BIT 4.9                  //Include previous result
SET_TMP_BIT 4.9                 //save partial Run2

                                //HMI Fallback NA

                                //3wire latch
AND_NOT_TMP_BIT 4.13            //NOT Swapping
AND_NOT_TMP_BIT 12.9            //NOT Mode Change 2
SET_TMP_BIT 4.9                 //save final Run 2

//
```

Structured Text Program (cont'd)

```
// Set Outputs to IMPR
//
                          //Process Output 1
LOAD_TMP_BIT 4.8          //Run1
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
SET_BIT 1200.12           //Output 1
SET_BIT 1200.9            //Aux 1 LED

                          //Process Output 2
LOAD_TMP_BIT 4.9          //Run2
AND_NOT_TMP_BIT 4.7       //NOT Stop 2
SET_BIT 1200.13           //Output 2
SET_BIT 1200.10           //Aux 2 LED

                          //Process other outputs
LOAD_BIT 455.3            //IMPR Alarm status
SET_BIT 1200.14           //Output 3 = Alarm
LOAD_BIT 455.2            //IMPR Fault status
SET_NOT_BIT 1200.15       //Output 4 = Fault
LOAD_BIT 457.4            //Reset Input LI5
SET_BIT 1200.2            //Logic Reset
LOAD_TMP_BIT 3.0          //PLC active
SET_BIT 1200.6            //Logic Local/Remote
LOAD_TMP_BIT 4.5          //Global Stop
SET_BIT 1200.11           //Stop LED

//
// Manage Power-UP Done
//
LOAD_NOT_TMP_BIT 4.5
OR_TMP_BIT 4.0
SET_TMP_BIT 12.12         //Power-up Done

// Clear PLC Control on Control Transfer
LOAD_TMP_BIT 4 0          //Control Source Transfer
AND_NOT_BIT 683 10        //NOT Bumpless
LOAD_K_REG 65532          //0xFFFC
AND_REG 704               //mask off Run1 and Run2
ON_SET_REG 704 54         //Run bits on Bump Control Change
```

Structured Text Program for 3-Wire Overload Mode

Overview

The structured text program for the 3-wire overload mode is defined below:

Structured Text Program

```
LOGIC_ID 3   // 3-WIRE OVERLOAD MODE
// Temp register allocation
// Temp 0 and Temp 1 as scratch
// Temp 2 as Requested Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
// Temp 3 as Active Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
// Temp  4 as state bits group 1
//       0=Control Transfer in process
//       1=LO1 PLC fallback value
//       2=L02 PLC fallback value
//       3=L01 HMI fallback value
//       4=L02 HMI fallback value
//       5=Global Stop
//       6=Stop1
//       7=Stop2
//       8=Run1
//       9=Run2
// Temp 5 as state bits group 2
//
//
// Temp 12 as INPUT History
//       1=PLC Run 1
//       2=PLC Run 2
//       3=HMI Run 1
//       4=HMI Run 2
//       5=TS Run 1
//       6=TS Run 2
//       7=Mode Change 1
//       8=
```

Structured Text Program (cont'd)

```
//       9=Mode Change 2
//       10=
//       11=Bumpless in Process
//       12=Power-up Done
//
// Temp 50+ as general status registers
// Temp 50  as ONSET status transition time value
// Temp 51  as ONSET status Low to High timer
// Temp 52  as ONSET status High to Low timer
// Temp 53  Latch
// Temp 54  as ONSET status  704 Run1-Run2
//
//Save Requested Control.in Temp 2
//
LOAD_BIT 683.8          //TS/HMI
SET_TMP_BIT 0.1         //Debounce TS/HMI in scratch
LOAD_BIT 457.5          //LI6
SET_TMP_BIT 0.0         //Debounce LI6 in scratch
SET_TMP_BIT 2.0         //PLC Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_TMP_BIT 0.1         //TS/HMI debounced
SET_TMP_BIT 2.1         //HMI Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_NOT_TMP_BIT 0.1     //TS/HMI debounced
SET_TMP_BIT 2.2         //TS Control
//
//Look for control transfer
//
LOAD_TMP_BIT 4.0        // Transfer in Process
SET_TMP_BIT 0.0         //save old Transfer in Process
LOAD_TMP_REG 2          //Requested Mode
COMP_TMP_REG 3, 1       //is it Active Mode
LOAD_NOT_TMP_BIT 1.2    //Not equal
SET_TMP_BIT 4.0         //Transfer in Process
//
```

Structured Text Program (cont'd)

```
//Manage Bump/Bumpless
//
LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_TMP_BIT 12.11   //NOT Bumpless in Process
SET_TMP_BIT 12.11       //Bumpless in Process (one scan)

LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_BIT 683.10      //Not bumpless
AND_NOT_TMP_BIT 0.0     //Look for Edge
SET_TMP_BIT 4.0         //Transfer in Process
SET_TMP_BIT 12.7        //Mode Change 1
SET_TMP_BIT 12.9        //Mode Change 2
//
// Save Active Control Mode in Temp Reg 3
//
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.0         //PLC requested
SET_TMP_BIT 3.0         //PLC active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.1         //HMI requested
SET_TMP_BIT 3.1         //HMI Active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.2         //TS requested
SET_TMP_BIT 3.2         //TS active
//
```

Structured Text Program (cont'd)

```
// Generate PLC Fallback Values
//
LOAD_REG 682            //PLC fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.12         //last LO1 command
SET_TMP_BIT 4.1         //LO1 PLC fallback
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.13         //last LO2 command
SET_TMP_BIT 4.2         //LO2 PLC fallback
                        //---STEP(1)--- no action needed
                        //---OFF(2)---- no action needed
COMP_K_REG 3, 0         //---ON(3)-----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.1          //logical or with previous value
SET_TMP_BIT 4.1         //LO1 PLC fallback
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.2          //logical or with previous value
SET_TMP_BIT 4.2         //LO2 PLC fallback
COMP_K_REG 4, 0         //---ON OFF(4)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.1          //logical or with previous value
SET_TMP_BIT 4.1         //LO1 PLC fallback
COMP_K_REG 5, 0         //---OFF ON(5)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.2          //logical or with previous value
SET_TMP_BIT 4.2         //LO2 PLC fallback

//
```

Structured Text Program (cont'd)

```
// Generate HMI Fallback Values
//
LOAD_REG 645            //HMI fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.12         //last LO1 command
SET_TMP_BIT 4.3         //LO1 HMI fallback
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.13         //last LO2 command
SET_TMP_BIT 4.4         //LO2 HMI fallback
                        //---STEP(1)--- no action needed
                        //---OFF(2)---- no action needed
COMP_K_REG 3, 0         //---ON(3)-----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.3          //logical or with previous value
SET_TMP_BIT 4.3         //LO1 HMI fallback
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.4          //logical or with previous value
SET_TMP_BIT 4.4         //LO2 HMI fallback
COMP_K_REG 4, 0         //---ON OFF(4)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.3          //logical or with previous value
SET_TMP_BIT 4.3         //LO1 HMI fallback
COMP_K_REG 5, 0         //---OFF ON(5)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.4          //logical or with previous value
SET_TMP_BIT 4.4         //LO2 HMI fallback

//
```

Structured Text Program (cont'd)

```
// Latch HMI Keypad info
//
LOAD_BIT 1020.12        //Aux 1
SET_TMP_BIT 13.12
LOAD_BIT 1020.13        //Aux 2
SET_TMP_BIT 13.13
LOAD_BIT 1020.14        //Stop
SET_TMP_BIT 13.14

//
// Generate Global Stop in Temp Reg 4.5
//
LOAD_TMP_BIT 13.14      //HMI Stop Key
OR_NOT_BIT 457.3        //NOT Stop
OR_BIT 456.5            //Load Shed
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_TMP_BIT 3.0    //NOT PLC active
AND_NOT_TMP_BIT 3.1     //NOT HMI active
AND_NOT_TMP_BIT 3.2     //NOT TS active
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save final Global Stop


//
//Latch comm loss values in scratch 0
//
LOAD_BIT 456.8          //PLC Comm Loss
SET_TMP_BIT 0.0         //save in scratch bit 0
LOAD_BIT 456.7          //HMI Comm Loss
SET_TMP_BIT 0.1         //save in scratch bit 1

//
```

Structured Text Program (cont'd)

```
// Generate Stop1 and Stop2 Commands
//
                          //Generate Stop1
LOAD_TMP_BIT 4.5          //Global Stop
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_TMP_BIT 0.0          //PLC Comm Loss from scratch
AND_TMP_BIT 3.0           //PLC active
AND_NOT_TMP_BIT 4.1       //NOT LO1 PLC fallback value
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_TMP_BIT 0.1          //HMI Comm Loss from scratch
AND_TMP_BIT 3.1           //HMI active
AND_NOT_TMP_BIT 4.3       //NOT LO1 HMI fallback value
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1

LOAD_TMP_BIT 3.0          //PLC active
AND_NOT_BIT 704.0         //NOT PLC Run1
AND_TMP_BIT 4.8           //Run 1
AND_NOT_TMP_BIT 0.0       //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save final Stop1
```

Structured Text Program (cont'd)

```
                          //Generate Stop2
LOAD_TMP_BIT 4.5          //Global Stop
SET_TMP_BIT 4.7           //save partial Stop7
LOAD_TMP_BIT 0.0          //PLC Comm Loss from scratch
AND_TMP_BIT 3.0           //PLC active
AND_NOT_TMP_BIT 4.2       //NOT LO2 PLC fallback value
OR_TMP_BIT 4.7            //Include partial Stop2
SET_TMP_BIT 4.7           //save partial Stop2
LOAD_TMP_BIT 0.1          //HMI Comm Loss from scratch
AND_TMP_BIT 3.1           //HMI active
AND_NOT_TMP_BIT 4.4       //NOT LO1 HMI fallback value
OR_TMP_BIT 4.7            //Include partial Stop2
SET_TMP_BIT 4.7           //save partial Stop2

LOAD_TMP_BIT 3.0          //PLC active
AND_NOT_BIT 704.1         // NOT PLC Run2
AND_TMP_BIT 4.9           //Run 2
AND_NOT_TMP_BIT 0.0       //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.7            //Include partial Stop2
SET_TMP_BIT 4.7           //save final Stop2

//
// Generate Run1 and Run2 Commands
//
                          //Generate Run 1
                          //PLC mode
LOAD_TMP_BIT 12.1         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 704.0            //PLC Network Run1
AND_TMP_BIT 12.12         //Power-up Done
AND_NOT_TMP_BIT 4.6       //NOT Stop1
SET_TMP_BIT 12.1          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.0           //PLC active
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
AND_NOT_TMP_BIT 0.0       //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1
```

Structured Text Program (cont'd)

```
                              //PLC Fallback
LOAD_TMP_BIT 4.1              //PLC fallback value
AND_TMP_BIT 3.0               //PLC active
AND_TMP_BIT 0.0               //PLC Comm Loss from scratch
OR_TMP_BIT 4.8                //Include previous result
SET_TMP_BIT 4.8               //save partial Run 1

                              //HMI Fallback NA

                              //3wire latch
AND_NOT_TMP_BIT 4.6           //NOT Stop 1
AND_NOT_TMP_BIT 4.13          //NOT Swapping
AND_NOT_TMP_BIT 12.7          //NOT Mode Change 1
SET_TMP_BIT 4.8               //save final Run 1

                              //Generate Run 2
                              //PLC mode
LOAD_TMP_BIT 12.2             //Input history
AND_NOT_TMP_BIT 12.11         //NOT Bumpless in Process
SET_TMP_BIT 12.0              //Save previous history
LOAD_BIT 704.1                //PLC Network Run2
AND_TMP_BIT 12.12             //Power-up Done
AND_NOT_TMP_BIT 4.7           //NOT Stop2
SET_TMP_BIT 12.2              //Save new history
AND_NOT_TMP_BIT 12.0          //NOT previous history
AND_TMP_BIT 3.0               //PLC active
AND_NOT_TMP_BIT 4.7           //NOT Stop2
AND_NOT_TMP_BIT 0.0           //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.9                //Include previous result
SET_TMP_BIT 4.9               //save partial Run2

                              //PLC Fallback
LOAD_TMP_BIT 4.2              //PLC fallback value
AND_TMP_BIT 3.0               //PLC active
AND_TMP_BIT 0.0               //PLC Comm Loss from scratch
OR_TMP_BIT 4.9                //Include previous result
SET_TMP_BIT 4.9               //save partial Run2
```

Structured Text Program (cont'd)

```
                                //HMI Fallback NA

                                //3wire latch
AND_NOT_TMP_BIT 4.7             //NOT Stop 2
AND_NOT_TMP_BIT 4.13            //NOT Swapping
AND_NOT_TMP_BIT 12.9            //NOT Mode Change 2
SET_TMP_BIT 4.9                 //save final Run 2

//
// Set Outputs to IMPR
//
                                //Process Output 1
LOAD_TMP_BIT 4.8                //Run1
AND_NOT_TMP_BIT 4.6             //NOT Stop 1
SET_BIT 1200.12                 //Output 1
SET_BIT 1200.9                  //Aux 1 LED

                                //Process Output 2
LOAD_TMP_BIT 4.9                //Run2
AND_NOT_TMP_BIT 4.7             //NOT Stop 2
SET_BIT 1200.13                 //Output 2
SET_BIT 1200.10                 //Aux 2 LED

                                //Process other outputs
LOAD_BIT 455.3                  //IMPR Alarm status
SET_BIT 1200.14                 //Output 3 = Alarm
LOAD_BIT 455.2                  //IMPR Fault status
SET_NOT_BIT 1200.15             //Output 4 = Fault
LOAD_BIT 457.4                  //Reset Input LI5
SET_BIT 1200.2                  //Logic Reset
LOAD_TMP_BIT 3.0                //PLC active
SET_BIT 1200.6                  //Logic Local/Remote
LOAD_TMP_BIT 4.6                //Stop 1
OR_TMP_BIT 4.7                  //Stop 2
SET_BIT 1200.11                 //Stop LED

//
```

Structured Text Program (cont'd)

```
// Manage Power-UP Done
//
LOAD_NOT_TMP_BIT 4.5
OR_TMP_BIT 4.0
SET_TMP_BIT 12.12       //Power-up Done

// Clear PLC Control on Control Transfer
LOAD_TMP_BIT 4 0        //Control Source Transfer
AND_NOT_BIT 683 10      //NOT Bumpless
LOAD_K_REG 65532        //0xFFFC
AND_REG 704             //mask off Run1 and Run2
ON_SET_REG 704 54       //Run bits on Bump Control Change
```

Structured Text Program for 2-Wire Independent Mode

Overview

The structured text program for the 2-wire independent mode is defined below:

Structured Text Program

```
LOGIC_ID 4  // 2-WIRE INDEPENDENT MODE
// Temp register allocation
// Temp 0 and Temp 1 as scratch
// Temp 2 as Requested Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
// Temp 3 as Active Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
// Temp  4 as state bits group 1
//       0=Control Transfer in process
//       1=LO1 PLC fallback value
//       2=L02 PLC fallback value
//       3=L01 HMI fallback value
//       4=L02 HMI fallback value
//       5=Global Stop
//       6=Stop1
//       7=Stop2
//       8=Run1
//       9=Run2
// Temp 5 as state bits group 2
//
// Temp 12 as INPUT History
//       1=PLC Run 1
//       2=PLC Run 2
//       3=HMI Run 1
//       4=HMI Run 2
//       5=TS Run 1
//       6=TS Run 2
//       7=Mode Change 1
//       8=
```

Structured Text Program (cont'd)

```
//       9=Mode Change 2
//       10=
//       11=Bumpless in Process
//       12=Power-up Done
//
// Temp 50+ as general status registers
// Temp 50  as ONSET status transition time value
// Temp 51  as ONSET status Low to High timer
// Temp 52  as ONSET status High to Low timer
// Temp 53  Latch
// Temp 54  as ONSET status  704 Run1-Run2
//
//
//Save Requested Control.in Temp 2
//
LOAD_BIT 683.8          //TS/HMI
SET_TMP_BIT 0.1         //Debounce TS/HMI in scratch
LOAD_BIT 457.5          //LI6
SET_TMP_BIT 0.0         //Debounce LI6 in scratch
SET_TMP_BIT 2.0         //PLC Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_TMP_BIT 0.1         //TS/HMI debounced
SET_TMP_BIT 2.1         //HMI Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_NOT_TMP_BIT 0.1     //TS/HMI debounced
SET_TMP_BIT 2.2         //TS Control
//
//Look for control transfer
//
LOAD_TMP_BIT 4.0        // Transfer in Process
SET_TMP_BIT 0.0         //save old Transfer in Process
LOAD_TMP_REG 2          //Requested Mode
COMP_TMP_REG 3, 1       //is it Active Mode
LOAD_NOT_TMP_BIT 1.2    //Not equal
SET_TMP_BIT 4.0         //Transfer in Process
//
```

Structured Text Program (cont'd)

```
//Manage Bump/Bumpless
//
LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_TMP_BIT 12.11   //NOT Bumpless in Process
SET_TMP_BIT 12.11       //Bumpless in Process (one scan)

LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_BIT 683.10      //Not bumpless
AND_NOT_TMP_BIT 0.0     //Look for Edge
SET_TMP_BIT 4.0         //Transfer in Process
SET_TMP_BIT 12.7        //Mode Change 1
SET_TMP_BIT 12.9        //Mode Change 2
//
// Save Active Control Mode in Temp Reg 3
//
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.0         //PLC requested
SET_TMP_BIT 3.0         //PLC active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.1         //HMI requested
SET_TMP_BIT 3.1         //HMI Active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.2         //TS requested
SET_TMP_BIT 3.2         //TS active

//
```

Structured Text Program (cont'd)

```
// Generate PLC Fallback Values
//
LOAD_REG 682              //PLC fallback mode
COMP_K_REG 0, 0           //---HOLD(0)---
LOAD_TMP_BIT 0 2          //equal
AND_BIT 1200.12           //last PLC run1 command
SET_TMP_BIT 4.1           //LO1 PLC fallback
LOAD_TMP_BIT 0 2          //equal
AND_BIT 1200.13           //last PLC run2 command
SET_TMP_BIT 4.2           //LO2 PLC fallback
                          //---STEP(1)--- no action needed
                          //---OFF(2)---- no action needed
COMP_K_REG 3, 0           //---ON(3)-----
LOAD_K_BIT 1              //fallback to ON
AND_TMP_BIT 0 2           //equal
OR_TMP_BIT 4.1            //logical or with previous value
SET_TMP_BIT 4.1           //LO1 PLC fallback
LOAD_K_BIT 1              //fallback to ON
AND_TMP_BIT 0 2           //equal
OR_TMP_BIT 4.2            //logical or with previous value
SET_TMP_BIT 4.2           //LO2 PLC fallback
COMP_K_REG 4, 0           //---ON OFF(4)----
LOAD_K_BIT 1              //fallback to ON
AND_TMP_BIT 0 2           //equal
OR_TMP_BIT 4.1            //logical or with previous value
SET_TMP_BIT 4.1           //LO1 PLC fallback
COMP_K_REG 5, 0           //---OFF ON(5)----
LOAD_K_BIT 1              //fallback to ON
AND_TMP_BIT 0 2           //equal
OR_TMP_BIT 4.2            //logical or with previous value
SET_TMP_BIT 4.2           //LO2 PLC fallback

//
```

Structured Text Program (cont'd)

```
// Generate HMI Fallback Values
//
LOAD_REG 645            //HMI fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.12         //last HMI run1 command
SET_TMP_BIT 4.3         //LO1 HMI fallback
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.13         //last HMI run2 command
SET_TMP_BIT 4.4         //LO2 HMI fallback
                        //---STEP(1)--- no action needed
                        //---OFF(2)---- no action needed
COMP_K_REG 3, 0         //---ON(3)-----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.3          //logical or with previous value
SET_TMP_BIT 4.3         //LO1 HMI fallback
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.4          //logical or with previous value
SET_TMP_BIT 4.4         //LO2 HMI fallback
COMP_K_REG 4, 0         //---ON OFF(4)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.3          //logical or with previous value
SET_TMP_BIT 4.3         //LO1 HMI fallback
COMP_K_REG 5, 0         //---OFF ON(5)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.4          //logical or with previous value
SET_TMP_BIT 4.4         //LO2 HMI fallback

//
```

Structured Text Program (cont'd)

```
// Latch HMI Keypad info
//
LOAD_BIT 1020.12        //Aux 1
SET_TMP_BIT 13.12
LOAD_BIT 1020.13        //Aux 2
SET_TMP_BIT 13.13
LOAD_BIT 1020.14        //Stop
SET_TMP_BIT 13.14


//
// Generate Global Stop in Temp Reg 4.5
//
LOAD_TMP_BIT 3.1        //HMI Active
AND_BIT 455.2           //IMPR Fault status
OR_TMP_BIT 13.14        //HMI Stop Key
OR_BIT 456.5            //Load Shed
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_TMP_BIT 3.0    //NOT PLC active
AND_NOT_TMP_BIT 3.1     //NOT HMI active
AND_NOT_TMP_BIT 3.2     //NOT TS active
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save final Global Stop


//
//Latch comm loss values in scratch 0
//
LOAD_BIT 456.8          //PLC Comm Loss
SET_TMP_BIT 0.0         //save in scratch bit 0
LOAD_BIT 456.7          //HMI Comm Loss
SET_TMP_BIT 0.1         //save in scratch bit 1

//
```

Structured Text Program (cont'd)

```
// Generate Stop1 and Stop2 Commands
//
                          //Generate Stop1
LOAD_TMP_BIT 4.5          //Global Stop
OR_BIT 453.1              //Diag Fault 1
OR_BIT 453.2              //Diag Fault 2
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_NOT_BIT 1200.12      //NOT alread on
AND_BIT 456.4             //Rapid Cycle
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_TMP_BIT 0.0          //PLC Comm Loss from scratch
AND_TMP_BIT 3.0           //PLC active
AND_NOT_TMP_BIT 4.1       //NOT LO1 PLC fallback value
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_TMP_BIT 0.1          //HMI Comm Loss from scratch
AND_TMP_BIT 3.1           //HMI active
AND_NOT_TMP_BIT 4.3       //NOT LO1 HMI fallback value
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1

LOAD_TMP_BIT 3.0          //PLC active
AND_NOT_BIT 704.0         //NOT PLC Run1
AND_TMP_BIT 4.8           //Run 1
AND_NOT_TMP_BIT 0.0       //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1

LOAD_TMP_BIT 3.1          //HMI active
AND_NOT_TMP_BIT 13.12     //NOT HMI Run 1
AND_TMP_BIT 4.8           //Run 1
AND_NOT_TMP_BIT 0.1       //NOT HMI Comm Loss from scratch
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1
```

Structured Text Program (cont'd)

```
LOAD_TMP_BIT 3.2          //TS active
AND_NOT_BIT 457.0         //NOT TS Run 1
AND_TMP_BIT 4.8           //Run 1
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save final Stop1

                          //Generate Stop2
LOAD_TMP_BIT 4.5          //Global Stop
SET_TMP_BIT 4.7           //save partial Stop7
LOAD_TMP_BIT 0.0          //PLC Comm Loss from scratch
AND_TMP_BIT 3.0           //PLC active
AND_NOT_TMP_BIT 4.2       //NOT LO2 PLC fallback value
OR_TMP_BIT 4.7            //Include partial Stop2
SET_TMP_BIT 4.7           //save partial Stop2
LOAD_TMP_BIT 0.1          //HMI Comm Loss from scratch
AND_TMP_BIT 3.1           //HMI active
AND_NOT_TMP_BIT 4.4       //NOT LO1 HMI fallback value
OR_TMP_BIT 4.7            //Include partial Stop2
SET_TMP_BIT 4.7           //save partial Stop2

LOAD_TMP_BIT 3.0          //PLC active
AND_NOT_BIT 704.1         // NOT PLC Run2
AND_TMP_BIT 4.9           //Run 2
AND_NOT_TMP_BIT 0.0       //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.7            //Include partial Stop2
SET_TMP_BIT 4.7           //save partial Stop2

LOAD_TMP_BIT 3.1          //HMI active
AND_NOT_TMP_BIT 13.13     //NOT HMI Run 2
AND_TMP_BIT 4.9           //Run 2
AND_NOT_TMP_BIT 0.1       //NOT HMI Comm Loss from scratch
OR_TMP_BIT 4.7            //Include partial Stop2
SET_TMP_BIT 4.7           //save partial Stop2

LOAD_TMP_BIT 3.2          //TS active
AND_NOT_BIT 457.1         //NOT TS Run 2
AND_TMP_BIT 4.9           //Run 2
OR_TMP_BIT 4.7            //Include partial Stop2
SET_TMP_BIT 4.7           //save final Stop2
//
```

Structured Text Program (cont'd)

```
// Generate Run1 and Run2 Commands
//
                          //Generate Run 1
                          //PLC mode
LOAD_TMP_BIT 12.1         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 704.0            //PLC Network Run1
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.1          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.0           //PLC active
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

                          //HMI mode
LOAD_TMP_BIT 12.3         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
SET_TMP_BIT 12.0          //Save previous history
LOAD_TMP_BIT 13.12        //HMI Run1
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.3          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.1           //HMI active
AND_NOT_TMP_BIT 4.12      //Lockout Timer
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

                          //TS mode
LOAD_TMP_BIT 12.5         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 457.0            //LI1
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.5          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.2           //TS active
AND_NOT_TMP_BIT 4.12      //Lockout Timer
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1
```

Structured Text Program (cont'd)

```
                              //PLC Fallback
LOAD_TMP_BIT 4.1              //PLC fallback value
AND_TMP_BIT 3.0               //PLC active
AND_TMP_BIT 0.0               //PLC Comm Loss from scratch
OR_TMP_BIT 4.8                //Include previous result
SET_TMP_BIT 4.8               //save partial Run 1

                              //HMI Fallback
LOAD_TMP_BIT 4.3              //HMI fallback value
AND_TMP_BIT 3.1               //HMI active
AND_TMP_BIT 0.1               //HMI Comm Loss from scratch
OR_TMP_BIT 4.8                //Include previous result
SET_TMP_BIT 4.8               //save partial Run 1

                              //3wire latch
AND_NOT_TMP_BIT 4.13          //NOT Swapping
AND_NOT_TMP_BIT 12.7          //NOT Mode Change 1
SET_TMP_BIT 4.8               //save final Run 1

                              //Generate Run 2
                              //PLC mode
LOAD_TMP_BIT 12.2             //Input history
AND_NOT_TMP_BIT 12.11         //NOT Bumpless in Process
SET_TMP_BIT 12.0              //Save previous history
LOAD_BIT 704.1                //PLC Network Run2
AND_TMP_BIT 12.12             //Power-up Done
SET_TMP_BIT 12.2              //Save new history
AND_NOT_TMP_BIT 12.0          //NOT previous history
AND_TMP_BIT 3.0               //PLC active
OR_TMP_BIT 4.9                //Include previous result
SET_TMP_BIT 4.9               //save partial Run2
```

Structured Text Program (cont'd)

```
                          //HMI mode
LOAD_TMP_BIT 12.4         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
SET_TMP_BIT 12.0          //Save previous history
LOAD_TMP_BIT 13.13        //HMI Run2
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.4          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.1           //HMI active
AND_NOT_TMP_BIT 4.12      //Lockout Timer
OR_TMP_BIT 4.9            //Include previous result
SET_TMP_BIT 4.9           //save partial Run2

                          //TS mode
LOAD_TMP_BIT 12.6         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 457.1            //LI2
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.6          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.2           //TS active
AND_NOT_TMP_BIT 4.12      //Lockout Timer
OR_TMP_BIT 4.9            //Include previous result
SET_TMP_BIT 4.9           //save partial Run2

                          //PLC Fallback
LOAD_TMP_BIT 4.2          //PLC fallback value
AND_TMP_BIT 3.0           //PLC active
AND_TMP_BIT 0.0           //PLC Comm Loss from scratch
OR_TMP_BIT 4.9            //Include previous result
SET_TMP_BIT 4.9           //save partial Run2

                          //HMI Fallback
LOAD_TMP_BIT 4.4          //HMI fallback value
AND_TMP_BIT 3.1           //HMI active
AND_TMP_BIT 0.1           //HMI Comm Loss from scratch
OR_TMP_BIT 4.9            //Include previous result
SET_TMP_BIT 4.9           //save partial Run2
```

Structured Text Program (cont'd)

```
                          //3wire latch
AND_NOT_TMP_BIT 4.13      //NOT Swapping
AND_NOT_TMP_BIT 12.9      //NOT Mode Change 2
SET_TMP_BIT 4.9           //save final Run 2

//
// Set Outputs to IMPR
//
                          //Process Output 1
LOAD_TMP_BIT 4.8          //Run1
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
SET_BIT 1200.12           //Output 1
SET_BIT 1200.9            //Aux 1 LED
SET_BIT 1200.0            //Motor Run
SET_NOT_BIT 1200.1        //Motor Stop

                          //Process Output 2
LOAD_TMP_BIT 4.9          //Run2
AND_NOT_TMP_BIT 4.7       //NOT Stop 2
SET_BIT 1200.13           //Output 2
SET_BIT 1200.10           //Aux 2 LED

                          //Process other outputs
LOAD_BIT 455.3            //IMPR Alarm status
SET_BIT 1200.14           //Output 3 = Alarm
LOAD_BIT 455.2            //IMPR Fault status
SET_NOT_BIT 1200.15       //Output 4 = Fault
LOAD_BIT 457.4            //Reset Input LI5
SET_BIT 1200.2            //Logic Reset
LOAD_TMP_BIT 3.0          //PLC active
SET_BIT 1200.6            //Logic Local/Remote

LOAD_NOT_BIT 1200.12      //NOT LO1 already on
AND_BIT 456.4             //Rapid Cycle
OR_TMP_BIT 4.5            //Global Stop
OR_BIT 453.1              //Diag Fault 1
OR_BIT 453.2              //Diag Fault 2
SET_BIT 1200.11           //Stop LED

//
```

Structured Text Program (cont'd)

```
// Manage Power-UP Done
//
LOAD_NOT_TMP_BIT 4.5
OR_TMP_BIT 4.0
SET_TMP_BIT 12.12       //Power-up Done

// Clear PLC Control on Control Transfer
LOAD_TMP_BIT 4 0        //Control Source Transfer
AND_NOT_BIT 683 10      //NOT Bumpless
LOAD_K_REG 65532        //0xFFFC
AND_REG 704             //mask off Run1 and Run2
ON_SET_REG 704 54       //Run bits on Bump Control Change
```

Structured Text Program for 3-Wire Independent Mode

Overview

The structured text program for the 3-wire independent mode is defined below:

Structured Text Program

```
LOGIC_ID 5  // 3-WIRE INDEPENDENT MODE
// Temp register allocation
// Temp 0 and Temp 1 as scratch
// Temp 2 as Requested Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
// Temp 3 as Active Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
// Temp  4 as state bits group 1
//       0=Control Transfer in process
//       1=LO1 PLC fallback value
//       2=L02 PLC fallback value
//       3=L01 HMI fallback value
//       4=L02 HMI fallback value
//       5=Global Stop
//       6=Stop1
//       7=Stop2
//       8=Run1
//       9=Run2
// Temp 5 as state bits group 2
//
// Temp 12 as INPUT History
//       1=PLC Run 1
//       2=PLC Run 2
//       3=HMI Run 1
//       4=HMI Run 2
//       5=TS Run 1
//       6=TS Run 2
//       7=Mode Change 1
//       8=
```

Structured Text Program (cont'd)

```
//       9=Mode Change 2
//       10=
//       11=Bumpless in Process
//       12=Power-up Done
//
// Temp 50+ as general status registers
// Temp 50  as ONSET status transition time value
// Temp 51  as ONSET status Low to High timer
// Temp 52  as ONSET status High to Low timer
// Temp 53  Latch
// Temp 54  as ONSET status  704 Run1-Run2
//
//
//Save Requested Control.in Temp 2
//
LOAD_BIT 683.8          //TS/HMI
SET_TMP_BIT 0.1         //Debounce TS/HMI in scratch
LOAD_BIT 457.5          //LI6
SET_TMP_BIT 0.0         //Debounce LI6 in scratch
SET_TMP_BIT 2.0         //PLC Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_TMP_BIT 0.1         //TS/HMI debounced
SET_TMP_BIT 2.1         //HMI Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_NOT_TMP_BIT 0.1     //TS/HMI debounced
SET_TMP_BIT 2.2         //TS Control
//
//Look for control transfer
//
LOAD_TMP_BIT 4.0        // Transfer in Process
SET_TMP_BIT 0.0         //save old Transfer in Process
LOAD_TMP_REG 2          //Requested Mode
COMP_TMP_REG 3, 1       //is it Active Mode
LOAD_NOT_TMP_BIT 1.2    //Not equal
SET_TMP_BIT 4.0         //Transfer in Process
//
```

Structured Text Program (cont'd)

```
//Manage Bump/Bumpless
//
LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_TMP_BIT 12.11   //NOT Bumpless in Process
SET_TMP_BIT 12.11       //Bumpless in Process (one scan)

LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_BIT 683.10      //Not bumpless
AND_NOT_TMP_BIT 0.0     //Look for Edge
SET_TMP_BIT 4.0         //Transfer in Process
SET_TMP_BIT 12.7        //Mode Change 1
SET_TMP_BIT 12.9        //Mode Change 2
//
// Save Active Control Mode in Temp Reg 3
//
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.0         //PLC requested
SET_TMP_BIT 3.0         //PLC active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.1         //HMI requested
SET_TMP_BIT 3.1         //HMI Active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.2         //TS requested
SET_TMP_BIT 3.2         //TS active
```

Structured Text Program (cont'd)

```
// Generate PLC Fallback Values
//
LOAD_REG 682            //PLC fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.12         //last PLC run1 command
SET_TMP_BIT 4.1         //LO1 PLC fallback
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.13         //last PLC run2 command
SET_TMP_BIT 4.2         //LO2 PLC fallback
                        //---STEP(1)--- no action needed
                        //---OFF(2)---- no action needed
COMP_K_REG 3, 0         //---ON(3)-----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.1          //logical or with previous value
SET_TMP_BIT 4.1         //LO1 PLC fallback
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.2          //logical or with previous value
SET_TMP_BIT 4.2         //LO2 PLC fallback
COMP_K_REG 4, 0         //---ON OFF(4)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.1          //logical or with previous value
SET_TMP_BIT 4.1         //LO1 PLC fallback
COMP_K_REG 5, 0         //---OFF ON(5)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.2          //logical or with previous value
SET_TMP_BIT 4.2         //LO2 PLC fallback

//
```

Structured Text Program (cont'd)

```
// Generate HMI Fallback Values
//
LOAD_REG 645              //HMI fallback mode
COMP_K_REG 0, 0           //---HOLD(0)---
LOAD_TMP_BIT 0 2          //equal
AND_BIT 1200.12           //last HMI run1 command
SET_TMP_BIT 4.3           //LO1 HMI fallback
LOAD_TMP_BIT 0 2          //equal
AND_BIT 1200.13           //last HMI run2 command
SET_TMP_BIT 4.4           //LO2 HMI fallback
                          //---STEP(1)--- no action needed
                          //---OFF(2)---- no action needed
COMP_K_REG 3, 0           //---ON(3)-----
LOAD_K_BIT 1              //fallback to ON
AND_TMP_BIT 0 2           //equal
OR_TMP_BIT 4.3            //logical or with previous value
SET_TMP_BIT 4.3           //LO1 HMI fallback
LOAD_K_BIT 1              //fallback to ON
AND_TMP_BIT 0 2           //equal
OR_TMP_BIT 4.4            //logical or with previous value
SET_TMP_BIT 4.4           //LO2 HMI fallback
COMP_K_REG 4, 0           //---ON OFF(4)----
LOAD_K_BIT 1              //fallback to ON
AND_TMP_BIT 0 2           //equal
OR_TMP_BIT 4.3            //logical or with previous value
SET_TMP_BIT 4.3           //LO1 HMI fallback
COMP_K_REG 5, 0           //---OFF ON(5)----
LOAD_K_BIT 1              //fallback to ON
AND_TMP_BIT 0 2           //equal
OR_TMP_BIT 4.4            //logical or with previous value
SET_TMP_BIT 4.4           //LO2 HMI fallback

//
```

Structured Text Program (cont'd)

```
// Latch HMI Keypad info
//
LOAD_BIT 1020.12        //Aux 1
SET_TMP_BIT 13.12
LOAD_BIT 1020.13        //Aux 2
SET_TMP_BIT 13.13
LOAD_BIT 1020.14        //Stop
SET_TMP_BIT 13.14

//
// Generate Global Stop in Temp Reg 4.5
//
LOAD_TMP_BIT 13.14      //HMI Stop Key
OR_BIT 456.5            //Load Shed
OR_NOT_BIT 457.3        //NOT LI4
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_TMP_BIT 3.0    //NOT PLC active
AND_NOT_TMP_BIT 3.1     //NOT HMI active
AND_NOT_TMP_BIT 3.2     //NOT TS active
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save final Global Stop


//
//Latch comm loss values in scratch 0
//
LOAD_BIT 456.8          //PLC Comm Loss
SET_TMP_BIT 0.0         //save in scratch bit 0
LOAD_BIT 456.7          //HMI Comm Loss
SET_TMP_BIT 0.1         //save in scratch bit 1

//
```

Structured Text Program (cont'd)

```
// Generate Stop1 and Stop2 Commands
//
                              //Generate Stop1
LOAD_TMP_BIT 4.5              //Global Stop
OR_NOT_TMP_BIT 12.12          //NOT Powerup Done
OR_BIT 453.1                  //Diag Fault 1
OR_BIT 453.2                  //Diag Fault 2
SET_TMP_BIT 4.6               //save partial Stop1
LOAD_NOT_BIT 1200.12          //NOT alread on
AND_BIT 456.4                 //Rapid Cycle
OR_TMP_BIT 4.6                //Include partial Stop1
SET_TMP_BIT 4.6               //save partial Stop1
LOAD_TMP_BIT 0.0              //PLC Comm Loss from scratch
AND_TMP_BIT 3.0               //PLC active
AND_NOT_TMP_BIT 4.1           //NOT LO1 PLC fallback value
OR_TMP_BIT 4.6                //Include partial Stop1
SET_TMP_BIT 4.6               //save partial Stop1
LOAD_TMP_BIT 0.1              //HMI Comm Loss from scratch
AND_TMP_BIT 3.1               //HMI active
AND_NOT_TMP_BIT 4.3           //NOT LO1 HMI fallback value
OR_TMP_BIT 4.6                //Include partial Stop1
SET_TMP_BIT 4.6               //save partial Stop1

LOAD_TMP_BIT 3.0              //PLC active
AND_NOT_BIT 704.0             //NOT PLC Run1
AND_TMP_BIT 4.8               //Run 1
AND_NOT_TMP_BIT 0.0           //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.6                //Include partial Stop1
SET_TMP_BIT 4.6               //save final Stop1
```

Structured Text Program (cont'd)

```
                                //Generate Stop2
LOAD_TMP_BIT 4.5                //Global Stop
OR_NOT_TMP_BIT 12.12            //NOT Powerup Done
SET_TMP_BIT 4.7                 //save partial Stop7
LOAD_TMP_BIT 0.0                //PLC Comm Loss from scratch
AND_TMP_BIT 3.0                 //PLC active
AND_NOT_TMP_BIT 4.2             //NOT LO2 PLC fallback value
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2
LOAD_TMP_BIT 0.1                //HMI Comm Loss from scratch
AND_TMP_BIT 3.1                 //HMI active
AND_NOT_TMP_BIT 4.4             //NOT LO1 HMI fallback value
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2

LOAD_TMP_BIT 3.0                //PLC active
AND_NOT_BIT 704.1               // NOT PLC Run2
AND_TMP_BIT 4.9                 //Run 2
AND_NOT_TMP_BIT 0.0             //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save final Stop2

//
```

Structured Text Program (cont'd)

```
// Generate Run1 and Run2 Commands
//
                          //Generate Run 1
                          //PLC mode
LOAD_TMP_BIT 12.1         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 704.0            //PLC Network Run1
AND_TMP_BIT 12.12         //Power-up Done
AND_NOT_BIT 456 4         //NOT Rapid Cycle
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
SET_TMP_BIT 12.1          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.0           //PLC active
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
AND_NOT_TMP_BIT 0.0       //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1
//HMI mode
LOAD_TMP_BIT 12.3         //Input history
SET_TMP_BIT 12.0          //Save previous history
LOAD_TMP_BIT 13.12        //HMI Run1
SET_TMP_BIT 12.3          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.1           //HMI active
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
AND_NOT_TMP_BIT 0.1       //NOT HMI Comm Loss from scratch
AND_NOT_TMP_BIT 4.12      //Lockout Timer
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1
```

Structured Text Program (cont'd)

```
                                //TS mode
LOAD_TMP_BIT 12.5               //Input history
SET_TMP_BIT 12.0                //Save previous history
LOAD_BIT 457.0                  //LI1
SET_TMP_BIT 12.5                //Save new history
AND_NOT_TMP_BIT 12.0            //NOT previous history
AND_TMP_BIT 3.2                 //TS active
AND_NOT_TMP_BIT 4.6             //NOT Stop 1
AND_NOT_TMP_BIT 4.12            //Lockout Timer
OR_TMP_BIT 4.8                  //Include previous result
SET_TMP_BIT 4.8                 //save partial Run1

                                //PLC Fallback
LOAD_TMP_BIT 4.1                //PLC fallback value
AND_TMP_BIT 3.0                 //PLC active
AND_TMP_BIT 0.0                 //PLC Comm Loss from scratch
OR_TMP_BIT 4.8                  //Include previous result
SET_TMP_BIT 4.8                 //save partial Run 1

                                //HMI Fallback
LOAD_TMP_BIT 4.3                //HMI fallback value
AND_TMP_BIT 3.1                 //HMI active
AND_TMP_BIT 0.1                 //HMI Comm Loss from scratch
OR_TMP_BIT 4.8                  //Include previous result
SET_TMP_BIT 4.8                 //save partial Run 1


//3wire latch
AND_NOT_TMP_BIT 4.6             //NOT Stop 1
AND_NOT_TMP_BIT 4.13            //NOT Swapping
AND_NOT_TMP_BIT 12.7            //NOT Mode Change 1
SET_TMP_BIT 4.8                 //save final Run 1
```

Structured Text Program (cont'd)

```
                                //Generate Run 2
                                //PLC mode
LOAD_TMP_BIT 12.2               //Input history
AND_NOT_TMP_BIT 12.11           //NOT Bumpless in Process
SET_TMP_BIT 12.0                //Save previous history
LOAD_BIT 704.1                  //PLC Network Run2
AND_TMP_BIT 12.12               //Power-up Done
AND_NOT_TMP_BIT 4.7             //NOT Stop 2
SET_TMP_BIT 12.2                //Save new history
AND_NOT_TMP_BIT 12.0            //NOT previous history
AND_TMP_BIT 3.0                 //PLC active
AND_NOT_TMP_BIT 4.7             //NOT Stop2
AND_NOT_TMP_BIT 0.0             //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.9                  //Include previous result
SET_TMP_BIT 4.9                 //save partial Run2

                                //HMI mode
LOAD_TMP_BIT 12.4               //Input history
SET_TMP_BIT 12.0                //Save previous history
LOAD_TMP_BIT 13.13              //HMI Run2
SET_TMP_BIT 12.4                //Save new history
AND_NOT_TMP_BIT 12.0            //NOT previous history
AND_TMP_BIT 3.1                 //HMI active
AND_NOT_TMP_BIT 4.7             //NOT Stop 2
AND_NOT_TMP_BIT 0.1             //NOT HMI Comm Loss from scratch
AND_NOT_TMP_BIT 4.12            //Lockout Timer
OR_TMP_BIT 4.9                  //Include previous result

SET_TMP_BIT 4.9                 //save partial Run2
```

Structured Text Program (cont'd)

```
//TS mode
LOAD_TMP_BIT 12.6       //Input history
SET_TMP_BIT 12.0        //Save previous history
LOAD_BIT 457.1          //LI2
SET_TMP_BIT 12.6        //Save new history
AND_NOT_TMP_BIT 12.0    //NOT previous history
AND_TMP_BIT 3.2         //TS active
AND_NOT_TMP_BIT 4.7     //NOT Stop 2
AND_NOT_TMP_BIT 4.12    //Lockout Timer
OR_TMP_BIT 4.9          //Include previous result
SET_TMP_BIT 4.9         //save partial Run2

                        //PLC Fallback
LOAD_TMP_BIT 4.2        //PLC fallback value
AND_TMP_BIT 3.0         //PLC active
AND_TMP_BIT 0.0         //PLC Comm Loss from scratch
OR_TMP_BIT 4.9          //Include previous result
SET_TMP_BIT 4.9         //save partial Run2

                        //HMI Fallback
LOAD_TMP_BIT 4.4        //HMI fallback value
AND_TMP_BIT 3.1         //HMI active
AND_TMP_BIT 0.1         //HMI Comm Loss from scratch
OR_TMP_BIT 4.9          //Include previous result
SET_TMP_BIT 4.9         //save partial Run2

                        //3wire latch
AND_NOT_TMP_BIT 4.7     //NOT Stop 2
AND_NOT_TMP_BIT 4.13    //NOT Swapping
AND_NOT_TMP_BIT 12.9    //NOT Mode Change 2
SET_TMP_BIT 4.9         //save final Run 2

//
```

Structured Text Program (cont'd)

```
// Set Outputs to IMPR
//
                              //Process Output 1
LOAD_TMP_BIT 4.8              //Run1
AND_NOT_TMP_BIT 4.6           //NOT Stop 1
SET_BIT 1200.12               //Output 1
SET_BIT 1200.9                //Aux 1 LED
SET_BIT 1200.0                //Motor Run
SET_NOT_BIT 1200.1            //Motor Stop

                              //Process Output 2
LOAD_TMP_BIT 4.9              //Run2
AND_NOT_TMP_BIT 4.7           //NOT Stop 2
SET_BIT 1200.13               //Output 2
SET_BIT 1200.10               //Aux 2 LED

                              //Process other outputs
LOAD_BIT 455.3                //IMPR Alarm status
SET_BIT 1200.14               //Output 3 = Alarm
LOAD_BIT 455.2                //IMPR Fault status
SET_NOT_BIT 1200.15           //Outpur 4 = Fault
LOAD_BIT 457.4                //Reset Input LI5
SET_BIT 1200.2                //Logic Reset
LOAD_TMP_BIT 3.0              //PLC active
SET_BIT 1200.6                //Logic Local/Remote
LOAD_TMP_BIT 4.6              //Stop 1
OR_TMP_BIT 4.7                //Stop 2
SET_BIT 1200.11               //Stop LED

//
```

Structured Text Program (cont'd)

```
// Manage Power-UP Done
//
LOAD_NOT_TMP_BIT 4.5
OR_TMP_BIT 4.0
SET_TMP_BIT 12.12       //Power-up Done

// Clear PLC Control on Control Transfer
LOAD_TMP_BIT 4 0        //Control Source Transfer
AND_NOT_BIT 683 10      //NOT Bumpless
LOAD_K_REG 65532        //0xFFFC
AND_REG 704             //mask off Run1 and Run2
ON_SET_REG 704 54       //Run bits on Bump Control Change
```

Structured Text Program for 2-Wire Reverser Mode

Overview

The structured text program for the 2-wire reverser mode is defined below:

Structured Text Program

```
LOGIC_ID 6  //2-WIRE REVERSER MODE
// Temp register allocation
// Temp 0 and Temp 1 as scratch
// Temp 2 as Requested Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
//
// Temp 3 as Active Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
//
// Temp  4 as state bits group 1
//       0=Control Transfer in process
//       1=LO1 PLC fallback value
//       2=L02 PLC fallback value
//       3=L01 HMI fallback value
//       4=L02 HMI fallback value
//       5=Global Stop
//       6=Stop1
//       7=Stop2
//       8=Run1
//       9=Run2
//      10=Forward
//      11=Reverse
//      12=Reversing Timer Active
//
// Temp 5 as state bits group 2
//
// Temp 9, 10, 11 as Forward Reverse Timer
//
```

Structured Text Program (cont'd)

```
// Temp 12 as INPUT History
//       1=PLC Run 1
//       2=PLC Run 2
//       3=HMI Run 1
//       4=HMI Run 2
//       5=TS Run 1
//       6=TS Run 2
//       7=Mode Change 1
//       8=
//       9=Mode Change 2
//       10=
//       11=Bumpless in Process
//       12=Power-up Done
//
// Temp 50+ as general status registers
// Temp 50  as ONSET status transition time value
// Temp 51  as ONSET status Low to High timer
// Temp 52  as ONSET status High to Low timer
// Temp 53  Latch
// Temp 54  as ONSET status  704 Run1-Run2
//
//Save Requested Control.in Temp 2
//
LOAD_BIT 683.8          //TS/HMI
SET_TMP_BIT 0.1         //Debounce TS/HMI in scratch
LOAD_BIT 457.5          //LI6
SET_TMP_BIT 0.0         //Debounce LI6 in scratch
SET_TMP_BIT 2.0         //PLC Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_TMP_BIT 0.1         //TS/HMI debounced
SET_TMP_BIT 2.1         //HMI Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_NOT_TMP_BIT 0.1     //TS/HMI debounced
SET_TMP_BIT 2.2         //TS Control
//
```

Structured Text Program (cont'd)

```
//Look for control transfer
//
LOAD_TMP_BIT 4.0        // Transfer in Process
SET_TMP_BIT 0.0         //save old Transfer in Process
LOAD_TMP_REG 2          //Requested Mode
COMP_TMP_REG 3, 1       //is it Active Mode
LOAD_NOT_TMP_BIT 1.2    //Not equal
SET_TMP_BIT 4.0         //Transfer in Process
//
//Manage Bump/Bumpless
//
LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_TMP_BIT 12.11   //NOT Bumpless in Process
AND_BIT 683.10          //Bumpless
SET_TMP_BIT 12.11       //Bumpless in Process (one scan)

LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_BIT 683.10      //Not bumpless
AND_NOT_TMP_BIT 0.0     //Look for Edge
SET_TMP_BIT 4.0         //Transfer in Process
SET_TMP_BIT 12.7        //Mode Change 1
SET_TMP_BIT 12.9        //Mode Change 2

//
// Save Active Control Mode in Temp Reg 3
//
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.0         //PLC requested
SET_TMP_BIT 3.0         //PLC active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.1         //HMI requested
SET_TMP_BIT 3.1         //HMI Active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.2         //TS requested
SET_TMP_BIT 3.2         //TS active
//
```

Structured Text Program (cont'd)

```
// Generate PLC Fallback Values
//
LOAD_REG 682            //PLC fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.12         //last LO1 command
SET_TMP_BIT 4.1         //LO1 PLC fallback
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.13         //last LO2 command
SET_TMP_BIT 4.2         //LO2 PLC fallback
                        //---STEP(1)--- no action needed
                        //---OFF(2)---- no action needed
                        //---ON(3)----- no action needed
COMP_K_REG 4, 0         //---ON OFF(4)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.1          //logical or with previous value
SET_TMP_BIT 4.1         //LO1 PLC fallback
COMP_K_REG 5, 0         //---OFF ON(5)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.2          //logical or with previous value
SET_TMP_BIT 4.2         //LO2 PLC fallback

//
```

Structured Text Program (cont'd)

```
// Generate HMI Fallback Values
//
LOAD_REG 645            //HMI fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.12         //last LO1 command
SET_TMP_BIT 4.3         //LO1 HMI fallback
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.13         //last LO2 command
SET_TMP_BIT 4.4         //LO2 HMI fallback
                        //---STEP(1)--- no action needed
                        //---OFF(2)---- no action needed
                        //---ON(3)----- no action needed
COMP_K_REG 4, 0         //---ON OFF(4)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.3          //logical or with previous value
SET_TMP_BIT 4.3         //LO1 HMI fallback
COMP_K_REG 5, 0         //---OFF ON(5)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.4          //logical or with previous value
SET_TMP_BIT 4.4         //LO2 HMI fallback

//
// Latch HMI Keypad info
//
LOAD_BIT 1020.12        //Aux 1
SET_TMP_BIT 13.12
LOAD_BIT 1020.13        //Aux 2
SET_TMP_BIT 13.13
LOAD_BIT 1020.14        //Stop
SET_TMP_BIT 13.14

//
```

Structured Text Program (cont'd)

```
// Generate Global Stop in Temp Reg 4.5
//
LOAD_TMP_BIT 3.1        //HMI Active
AND_BIT 455.2           //NOT IMPR Fault status
OR_TMP_BIT 13.14        //HMI Stop Key
OR_BIT 456.5            //Load Shed
OR_BIT 453.1            //Diag Fault 1
OR_BIT 453.2            //Diag Fault 2
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_TMP_BIT 3.0    //NOT PLC active
AND_NOT_TMP_BIT 3.1     //NOT HMI active
AND_NOT_TMP_BIT 3.2     //NOT TS active
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_BIT 1200.0     //NOT already on
AND_BIT 456.4           //Rapid Cycle
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save final Global Stop

//
//Latch comm loss values in scratch 0
//
LOAD_BIT 456.8          //PLC Comm Loss
SET_TMP_BIT 0.0         //save in scratch bit 0
LOAD_BIT 456.7          //HMI Comm Loss
SET_TMP_BIT 0.1         //save in scratch bit 1

//
```

Structured Text Program (cont'd)

```
// Generate Stop1 and Stop2 Commands
//
                              //Generate Stop1
LOAD_TMP_BIT 4.5              //Global Stop
SET_TMP_BIT 4.6               //save partial Stop1
LOAD_TMP_BIT 0.0              //PLC Comm Loss from scratch
AND_TMP_BIT 3.0               //PLC active
AND_NOT_TMP_BIT 4.1           //NOT LO1 PLC fallback value
OR_TMP_BIT 4.6                //Include partial Stop1
SET_TMP_BIT 4.6               //save partial Stop1
LOAD_TMP_BIT 0.1              //HMI Comm Loss from scratch
AND_TMP_BIT 3.1               //HMI active
AND_NOT_TMP_BIT 4.3           //NOT LO1 HMI fallback value
OR_TMP_BIT 4.6                //Include partial Stop1
SET_TMP_BIT 4.6               //save partial Stop1

LOAD_TMP_BIT 3.0              //PLC active
AND_NOT_BIT 704.0             //NOT PLC Run1
AND_TMP_BIT 4.8               //Run 1
AND_NOT_TMP_BIT 0.0           //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.6                //Include partial Stop1
SET_TMP_BIT 4.6               //save partial Stop1

LOAD_TMP_BIT 3.1              //HMI active
AND_NOT_TMP_BIT 13.12         //NOT HMI Run 1
AND_TMP_BIT 4.8               //Run 1
AND_NOT_TMP_BIT 0.1           //NOT HMI Comm Loss from scratch
OR_TMP_BIT 4.6                //Include partial Stop1
SET_TMP_BIT 4.6               //save partial Stop1

LOAD_TMP_BIT 3.2              //TS active
AND_NOT_BIT 457.0             //NOT TS Run 1
AND_TMP_BIT 4.8               //Run 1
OR_TMP_BIT 4.6                //Include partial Stop1
SET_TMP_BIT 4.6               //save final Stop1
```

Structured Text Program (cont'd)

```
                                //Generate Stop2
LOAD_TMP_BIT 4.5                //Global Stop
SET_TMP_BIT 4.7                 //save partial Stop7
LOAD_TMP_BIT 0.0                //PLC Comm Loss from scratch
AND_TMP_BIT 3.0                 //PLC active
AND_NOT_TMP_BIT 4.2             //NOT LO2 PLC fallback value
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2
LOAD_TMP_BIT 0.1                //HMI Comm Loss from scratch
AND_TMP_BIT 3.1                 //HMI active
AND_NOT_TMP_BIT 4.4             //NOT LO1 HMI fallback value
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2

LOAD_TMP_BIT 3.0                //PLC active
AND_NOT_BIT 704.1               // NOT PLC Run2
AND_TMP_BIT 4.9                 //Run 2

AND_NOT_TMP_BIT 0.0             //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2

LOAD_TMP_BIT 3.1                //HMI active
AND_NOT_TMP_BIT 13.13           //NOT HMI Run 2
AND_TMP_BIT 4.9                 //Run 2
AND_NOT_TMP_BIT 0.1             //NOT HMI Comm Loss from scratch
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2

LOAD_TMP_BIT 3.2                //TS active
AND_NOT_BIT 457.1               //NOT TS Run 2
AND_TMP_BIT 4.9                 //Run 2
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save final Stop2
```

Structured Text Program (cont'd)

```
// Generate Run1 and Run2 Commands
//
                          //Generate Run 1
                          //PLC mode
LOAD_TMP_BIT 12.1         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
AND_NOT_TMP_BIT 4.6       //NOT Stop1
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 704.0            //PLC Network Run1
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.1          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.0           //PLC active
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

                         //HMI mode
LOAD_TMP_BIT 12.3        //Input history
AND_NOT_TMP_BIT 12.11    //NOT Bumpless in Process
AND_NOT_TMP_BIT 4.6      //NOT Stop1
SET_TMP_BIT 12.0         //Save previous history
LOAD_TMP_BIT 13.12       //HMI Run1
AND_TMP_BIT 12.12        //Power-up Done
SET_TMP_BIT 12.3         //Save new history
AND_NOT_TMP_BIT 12.0     //NOT previous history
AND_TMP_BIT 3.1          //HMI active
OR_TMP_BIT 4.8           //Include previous result
SET_TMP_BIT 4.8          //save partial Run1
```

Structured Text Program (cont'd)

```
                          //TS mode
LOAD_TMP_BIT 12.5         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
AND_NOT_TMP_BIT 4.6       //NOT Stop1
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 457.0            //LI1
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.5          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.2           //TS active
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

                          //PLC Fallback
LOAD_TMP_BIT 4.1          //PLC fallback value
AND_TMP_BIT 3.0           //PLC active
AND_TMP_BIT 0.0           //PLC Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run 1

                          //HMI Fallback
LOAD_TMP_BIT 4.3          //HMI fallback value
AND_TMP_BIT 3.1           //HMI active
AND_TMP_BIT 0.1           //HMI Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run 1

                          //3wire latch
AND_NOT_TMP_BIT 12.7      //NOT Mode Change 1
SET_TMP_BIT 4.8           //save final Run 1
```

Structured Text Program (cont'd)

```
                          //Generate Run 2
                          //PLC mode
LOAD_TMP_BIT 12.2         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
AND_NOT_TMP_BIT 4.7       //NOT Stop2
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 704.1            //PLC Network Run2
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.2          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.0           //PLC active
OR_TMP_BIT 4.9            //Include previous result
SET_TMP_BIT 4.9           //save partial Run2

                          //HMI mode
LOAD_TMP_BIT 12.4         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
AND_NOT_TMP_BIT 4.7       //NOT Stop2
SET_TMP_BIT 12.0          //Save previous history
LOAD_TMP_BIT 13.13        //HMI Run2
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.4          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.1           //HMI active
OR_TMP_BIT 4.9            //Include previous result
SET_TMP_BIT 4.9           //save partial Run2

                          //TS mode
LOAD_TMP_BIT 12.6         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
AND_NOT_TMP_BIT 4.7       //NOT Stop2
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 457.1            //LI2
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.6          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.2           //TS active
OR_TMP_BIT 4.9            //Include previous result
SET_TMP_BIT 4.9           //save partial Run2
```

Structured Text Program (cont'd)

```
                                //PLC Fallback
LOAD_TMP_BIT 4.2                //PLC fallback value
AND_TMP_BIT 3.0                 //PLC active
AND_TMP_BIT 0.0                 //PLC Comm Loss from scratch
OR_TMP_BIT 4.9                  //Include previous result
SET_TMP_BIT 4.9                 //save partial Run2

                                //HMI Fallback
LOAD_TMP_BIT 4.4                //HMI fallback value
AND_TMP_BIT 3.1                 //HMI active
AND_TMP_BIT 0.1                 //HMI Comm Loss from scratch
OR_TMP_BIT 4.9                  //Include previous result
SET_TMP_BIT 4.9                 //save partial Run2

                                //3wire latch
AND_NOT_TMP_BIT 12.9            //NOT Mode Change 2
SET_TMP_BIT 4.9                 //save final Run 2

//
```

Structured Text Program (cont'd)

```
//Manage forward-reverse timer
//

LOAD_REG 541            //Forward-Reverse Time value
LOAD_K_BIT 1            //Force a rising edge
SET_NOT_TMP_BIT 51.3    //Force history bit off
ON_SET_TMP_REG 9, 51    //Timer Value

LOAD_NOT_TMP_BIT 4.8    //NOT Run1
OR_TMP_BIT 4.6          //Stop 1
AND_TMP_BIT 4.10        //Forward
SET_TMP_BIT 0.0         //save partial result in scratch

LOAD_NOT_TMP_BIT 4.9    //NOT Run2
OR_TMP_BIT 4.7          //Stop 2
AND_TMP_BIT 4.11        //Reverse
OR_TMP_BIT 0.0          //include partial result
SET_TMP_BIT 0.0         //save partial result in scratch

LOAD_NOT_TMP_BIT 4.10   //NOT Forward
AND_NOT_TMP_BIT 4.11    //NOT Reverse

AND_TMP_BIT 11.2        //already timing
OR_TMP_BIT 0.0          //include partial result
OR_NOT_TMP_BIT 12.12    //NOT Power-up Done
SET_TMP_BIT 11.0        //Enable Timer

TIMER_TENTHS 9,10,11    //Process forward-reverse timer

//
//update Lockout Timing flag
//

LOAD_TMP_BIT 11.0       //Enabled
AND_TMP_BIT 11.2        //timing
SET_TMP_BIT 4.12        //Reversing Timer Active

//
```

Structured Text Program (cont'd)

```
//Manage Forward and Reverse status bits
//

LOAD_NOT_TMP_BIT 4.12   //NOT Reversing Timer Active
OR_TMP_BIT 52.0         //Last direction-forward
AND_TMP_BIT 4.8         //Run1
AND_NOT_TMP_BIT 4.6     //NOT Stop1
AND_NOT_TMP_BIT 4.11    //NOT Reverse
SET_TMP_BIT 4.10        //save Forward
SET_TMP_BIT 52.1        //set last direction=forward

LOAD_NOT_TMP_BIT 4.12   //NOT Reversing Timer Active
OR_NOT_TMP_BIT 52.0     //NOT last direction-forward
AND_TMP_BIT 4.9         //Run2
AND_NOT_TMP_BIT 4.7     //NOT Stop2
AND_NOT_TMP_BIT 4.10    //NOT Forward
SET_TMP_BIT 4.11        //save Reverse
SET_TMP_BIT 52.2        //set last direction=reverse

LATCH 52                //last direction latch


//
```

Structured Text Program (cont'd)

```
// Set Outputs to IMPR
//
                              //Process Output 1
LOAD_TMP_BIT 4.10             //Forward
SET_BIT 1200.12               //Output 1
SET_BIT 1200.9                //Aux 1 LED

                              //Process Output 2
LOAD_TMP_BIT 4.11             //Reverse
SET_BIT 1200.13               //Output 2
SET_BIT 1200.10               //Aux 2 LED
SET_BIT 1200.5                //Phase Reverse

LOAD_TMP_BIT 4.10             //Forward
OR_TMP_BIT 4.11               //Reverse
SET_BIT 1200.0                //Motor Run
SET_NOT_BIT 1200.1            //Motor Stop

LOAD_TMP_BIT 4.12             //Reversing Timer
SET_BIT 1200.4                //Transition Timer

                              //Process other outputs
LOAD_BIT 455.3                //IMPR Alarm status
SET_BIT 1200.14               //Output 3 = Alarm
LOAD_BIT 455.2                //IMPR Fault status
SET_NOT_BIT 1200.15           //Output 4 = Fault
LOAD_BIT 457.4                //Reset Input LI5
SET_BIT 1200.2                //Logic Reset
LOAD_TMP_BIT 3.0              //PLC active
SET_BIT 1200.6                //Logic Local/Remote
LOAD_TMP_BIT 4.5              //Global Stop
OR_TMP_BIT 4.12               //Reversing Timer Active
SET_BIT 1200.11               //Stop LED

//
```

Structured Text Program (cont'd)

```
// Manage Power-UP Done
//
LOAD_NOT_TMP_BIT 4.12   //Wait for power-up timer
OR_TMP_BIT 12.12        //Latch ON until next power-up
SET_TMP_BIT 12.12       //Power-up Done

// Clear PLC Control on Control Transfer
LOAD_TMP_BIT 4 0        //Control Source Transfer
AND_NOT_BIT 683 10      //NOT Bumpless
LOAD_K_REG 65532        //0xFFFC
AND_REG 704             //mask off Run1 and Run2
ON_SET_REG 704 54       //Run bits on Bump Control Change
```

Structured Text Program for 3-Wire Reverser Mode

Overview

The structured text program for the 3-wire reverser mode is defined below:

Structured Text Program

```
LOGIC_ID 7    //  3-WIRE REVERSER MODE
// Temp register allocation
// Temp 0 and Temp 1 as scratch
// Temp 2 as Requested Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
//
// Temp 3 as Active Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
//
// Temp  4 as state bits group 1
//       0=Control Transfer in process
//       1=LO1 PLC fallback value
//       2=L02 PLC fallback value
//       3=L01 HMI fallback value
//       4=L02 HMI fallback value
//       5=Global Stop
//       6=Stop1
//       7=Stop2
//       8=Run1
//       9=Run2
//      10=Forward
//      11=Reverse
//      12=Reversing Timer Active
//      13=Swapping
//      14=Last Direction
//      15=Two Wire Swap
//
//
```

Structured Text Program (cont'd)

```
// Temp 5 as state bits group 2
//
// Temp 9, 10, 11 as Forward Reverse Timer
//
// Temp 12 as INPUT History
//       1=PLC Run 1
//       2=PLC Run 2
//       3=HMI Run 1
//       4=HMI Run 2
//       5=TS Run 1
//       6=TS Run 2
//       7=Mode Change 1
//       8=
//       9=Mode Change 2
//       10=
//       11=Bumpless in Process
//       12=Power-up Done
//
// Temp 50+ as general status registers
// Temp 50  as ONSET status transition time value
// Temp 51  as ONSET status Low to High timer
// Temp 52  as ONSET status High to Low timer
// Temp 53  Latch
// Temp 54  as ONSET status  704 Run1-Run2
//
//Save Requested Control.in Temp 2
//
LOAD_BIT 683.8          //TS/HMI
SET_TMP_BIT 0.1         //Debounce TS/HMI in scratch
LOAD_BIT 457.5          //LI6
SET_TMP_BIT 0.0         //Debounce LI6 in scratch
SET_TMP_BIT 2.0         //PLC Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_TMP_BIT 0.1         //TS/HMI debounced
SET_TMP_BIT 2.1         //HMI Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_NOT_TMP_BIT 0.1     //TS/HMI debounced
SET_TMP_BIT 2.2         //TS Control
//
```

Structured Text Program (cont'd)

```
//Look for control transfer
//
LOAD_TMP_BIT 4.0        // Transfer in Process
SET_TMP_BIT 0.0         //save old Transfer in Process
LOAD_TMP_REG 2          //Requested Mode
COMP_TMP_REG 3, 1       //is it Active Mode
LOAD_NOT_TMP_BIT 1.2    //Not equal
SET_TMP_BIT 4.0         //Transfer in Process
//
//Manage Bump/Bumpless
//
LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_TMP_BIT 12.11   //NOT Bumpless in Process
SET_TMP_BIT 12.11       //Bumpless in Process (one scan)

LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_BIT 683.10      //Not bumpless
AND_NOT_TMP_BIT 0.0     //Look for Edge
SET_TMP_BIT 4.0         //Transfer in Process
SET_TMP_BIT 12.7        //Mode Change 1
SET_TMP_BIT 12.9        //Mode Change 2

//
// Save Active Control Mode in Temp Reg 3
//
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.0         //PLC requested
SET_TMP_BIT 3.0         //PLC active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.1         //HMI requested
SET_TMP_BIT 3.1         //HMI Active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.2         //TS requested
SET_TMP_BIT 3.2         //TS active
//
```

Structured Text Program (cont'd)

```
// Generate PLC Fallback Values
//
LOAD_REG 682            //PLC fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.12         //last LO1 command
SET_TMP_BIT 4.1         //LO1 PLC fallback
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.13         //last LO2 command
SET_TMP_BIT 4.2         //LO2 PLC fallback
                        //---STEP(1)--- no action needed
                        //---OFF(2)---- no action needed
                        //---ON(3)----- no action needed
COMP_K_REG 4, 0         //---ON OFF(4)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.1          //logical or with previous value
SET_TMP_BIT 4.1         //LO1 PLC fallback
COMP_K_REG 5, 0         //---OFF ON(5)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.2          //logical or with previous value
SET_TMP_BIT 4.2         //LO2 PLC fallback

//
```

Structured Text Program (cont'd)

```
// Generate HMI Fallback Values
//
LOAD_REG 645            //HMI fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.12         //last LO1 command
SET_TMP_BIT 4.3         //LO1 HMI fallback
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.13         //last LO2 command
SET_TMP_BIT 4.4         //LO2 HMI fallback
                        //---STEP(1)--- no action needed
                        //---OFF(2)---- no action needed
                        //---ON(3)----- no action needed
COMP_K_REG 4, 0         //---ON OFF(4)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.3          //logical or with previous value
SET_TMP_BIT 4.3         //LO1 HMI fallback
COMP_K_REG 5, 0         //---OFF ON(5)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.4          //logical or with previous value
SET_TMP_BIT 4.4         //LO2 HMI fallback

//
// Latch HMI Keypad info
//
LOAD_BIT 1020.12        //Aux 1
SET_TMP_BIT 13.12
LOAD_BIT 1020.13        //Aux 2
SET_TMP_BIT 13.13
LOAD_BIT 1020.14        //Stop
SET_TMP_BIT 13.14

//
```

Structured Text Program (cont'd)

```
// Generate Global Stop in Temp Reg 4.5
//
LOAD_TMP_BIT 13.14      //HMI Stop Key
OR_NOT_BIT 457.3        //NOT Stop
OR_BIT 456.5            //Load Shed
OR_BIT 453.1            //Diag Fault 1
OR_BIT 453.2            //Diag Fault 2
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_TMP_BIT 3.0    //NOT PLC active
AND_NOT_TMP_BIT 3.1     //NOT HMI active
AND_NOT_TMP_BIT 3.2     //NOT TS active
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_BIT 1200.0     //NOT already on
AND_BIT 456.4           //Rapid Cycle
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save final Global Stop

//
//Latch comm loss values in scratch 0
//
LOAD_BIT 456.8          //PLC Comm Loss
SET_TMP_BIT 0.0         //save in scratch bit 0
LOAD_BIT 456.7          //HMI Comm Loss
SET_TMP_BIT 0.1         //save in scratch bit 1

//
```

Structured Text Program (cont'd)

```
// Generate Stop1 and Stop2 Commands
//
                              //Generate Stop1
LOAD_TMP_BIT 4.5              //Global Stop
OR_NOT_TMP_BIT 12.12          //NOT Powerup Done
SET_TMP_BIT 4.6               //save partial Stop1
LOAD_TMP_BIT 0.0              //PLC Comm Loss from scratch
AND_TMP_BIT 3.0               //PLC active
AND_NOT_TMP_BIT 4.1           //NOT LO1 PLC fallback value
OR_TMP_BIT 4.6                //Include partial Stop1
SET_TMP_BIT 4.6               //save partial Stop1
LOAD_TMP_BIT 0.1              //HMI Comm Loss from scratch
AND_TMP_BIT 3.1               //HMI active
AND_NOT_TMP_BIT 4.3           //NOT LO1 HMI fallback value
OR_TMP_BIT 4.6                //Include partial Stop1
SET_TMP_BIT 4.6               //save partial Stop1

LOAD_TMP_BIT 3.0              //PLC active
AND_NOT_BIT 704.0             //NOT PLC Run1
AND_TMP_BIT 4.8               //Run 1
AND_NOT_TMP_BIT 0.0           //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.6                //Include partial Stop1
SET_TMP_BIT 4.6               //save final Stop1
```

Structured Text Program (cont'd)

```
//Generate Stop2
LOAD_TMP_BIT 4.5         //Global Stop
OR_NOT_TMP_BIT 12.12     //NOT Powerup Done
SET_TMP_BIT 4.7          //save partial Stop7
LOAD_TMP_BIT 0.0         //PLC Comm Loss from scratch
AND_TMP_BIT 3.0          //PLC active
AND_NOT_TMP_BIT 4.2      //NOT LO2 PLC fallback value
OR_TMP_BIT 4.7           //Include partial Stop2
SET_TMP_BIT 4.7          //save partial Stop2
LOAD_TMP_BIT 0.1         //HMI Comm Loss from scratch
AND_TMP_BIT 3.1          //HMI active
AND_NOT_TMP_BIT 4.4      //NOT LO1 HMI fallback value
OR_TMP_BIT 4.7           //Include partial Stop2
SET_TMP_BIT 4.7          //save partial Stop2

LOAD_TMP_BIT 3.0         //PLC active
AND_NOT_BIT 704.1        // NOT PLC Run2
AND_TMP_BIT 4.9          //Run 2
AND_NOT_TMP_BIT 0.0      //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.7           //Include partial Stop2
SET_TMP_BIT 4.7          //save final Stop2

//
```

Structured Text Program (cont'd)

```
// Generate Run1 and Run2 Commands
//
                          //Generate Run 1
                          //PLC mode
LOAD_TMP_BIT 12.1         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 704.0            //PLC Network Run1
AND_NOT_BIT 704.1         //NOT PLC Network Run2
AND_TMP_BIT 12.12         //Power-up Done
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
SET_TMP_BIT 12.1          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.0           //PLC active
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
AND_NOT_TMP_BIT 0.0       //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

//HMI mode
LOAD_TMP_BIT 12.3       //Input history
SET_TMP_BIT 12.0        //Save previous history
LOAD_TMP_BIT 13.12      //HMI Run1
SET_TMP_BIT 12.3        //Save new history
AND_NOT_TMP_BIT 12.0    //NOT previous history
AND_TMP_BIT 3.1         //HMI active
AND_NOT_TMP_BIT 4.6     //NOT Stop 1
AND_NOT_TMP_BIT 0.1     //NOT HMI Comm Loss from scratch
AND_NOT_TMP_BIT 4.12    //Lockout Timer
OR_TMP_BIT 4.8          //Include previous result
SET_TMP_BIT 4.8         //save partial Run1
```

Structured Text Program (cont'd)

```
                          //TS mode
LOAD_TMP_BIT 12.5         //Input history
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 457.0            //LI1
SET_TMP_BIT 12.5          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.2           //TS active
AND_NOT_TMP_BIT 4.6       //NOT Stop 1

//
SET_TMP_BIT 0.2
LOAD_TMP_BIT 4.14         //Last Dir-Forward
OR_NOT_TMP_BIT 4.12       //Lockout Timer
AND_TMP_BIT 0.2           //temp

//AND_NOT_TMP_BIT 4.12     //Lockout Timer
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

                          //PLC Fallback
LOAD_TMP_BIT 4.1          //PLC fallback value
AND_TMP_BIT 3.0           //PLC active
AND_TMP_BIT 0.0           //PLC Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run 1

                          //HMI Fallback
LOAD_TMP_BIT 4.3          //HMI fallback value
AND_TMP_BIT 3.1           //HMI active
AND_TMP_BIT 0.1           //HMI Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run 1

                          //3wire latch
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
AND_NOT_TMP_BIT 4.13      //NOT Swapping
AND_NOT_TMP_BIT 12.7      //NOT Mode Change 1
SET_TMP_BIT 4.8           //save final Run 1
```

Structured Text Program (cont'd)

```
                              //Generate Run 2
                              //PLC mode
LOAD_TMP_BIT 12.2             //Input history
AND_NOT_TMP_BIT 12.11         //NOT Bumpless in Process
SET_TMP_BIT 12.0              //Save previous history
LOAD_BIT 704.1                //PLC Network Run2
AND_NOT_BIT 704.0             //NOT PLC Network Run1
AND_TMP_BIT 12.12             //Power-up Done
AND_NOT_TMP_BIT 4.7           //NOT Stop 2
SET_TMP_BIT 12.2              //Save new history
AND_NOT_TMP_BIT 12.0          //NOT previous history
AND_TMP_BIT 3.0               //PLC active
AND_NOT_TMP_BIT 4.7           //NOT Stop2
AND_NOT_TMP_BIT 0.0           //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.9                //Include previous result
SET_TMP_BIT 4.9               //save partial Run2

                              //HMI mode
LOAD_TMP_BIT 12.4             //Input history
SET_TMP_BIT 12.0              //Save previous history
LOAD_TMP_BIT 13.13            //HMI Run2
SET_TMP_BIT 12.4              //Save new history
AND_NOT_TMP_BIT 12.0          //NOT previous history
AND_TMP_BIT 3.1               //HMI active
AND_NOT_TMP_BIT 4.7           //NOT Stop 2
AND_NOT_TMP_BIT 0.1           //NOT HMI Comm Loss from scratch
AND_NOT_TMP_BIT 4.12          //Lockout Timer
OR_TMP_BIT 4.9                //Include previous result
SET_TMP_BIT 4.9               //save partial Run2
```

Structured Text Program (cont'd)

```
                          //TS mode
LOAD_TMP_BIT 12.6         //Input history
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 457.1            //LI2
SET_TMP_BIT 12.6          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.2           //TS active
AND_NOT_TMP_BIT 4.7       //NOT Stop 2

SET_TMP_BIT 0.2           //temp
LOAD_NOT_TMP_BIT 4.14     //NOT Last Dir-Forward
OR_NOT_TMP_BIT 4.12       //Lockout Timer
AND_TMP_BIT 0.2           //temp
OR_TMP_BIT 4.9            //Include previous result
SET_TMP_BIT 4.9           //save partial Run2

                           //PLC Fallback
LOAD_TMP_BIT 4.2           //PLC fallback value
AND_TMP_BIT 3.0            //PLC active
AND_TMP_BIT 0.0            //PLC Comm Loss from scratch
OR_TMP_BIT 4.9             //Include previous result
SET_TMP_BIT 4.9            //save partial Run2

                           //HMI Fallback
LOAD_TMP_BIT 4.4           //HMI fallback value
AND_TMP_BIT 3.1            //HMI active
AND_TMP_BIT 0.1            //HMI Comm Loss from scratch
OR_TMP_BIT 4.9             //Include previous result
SET_TMP_BIT 4.9            //save partial Run2

                           //3wire latch
AND_NOT_TMP_BIT 4.7        //NOT Stop 2
AND_NOT_TMP_BIT 4.13       //NOT Swapping
AND_NOT_TMP_BIT 12.9       //NOT Mode Change 2
SET_TMP_BIT 4.9            //save final Run 2

//
```

Structured Text Program (cont'd)

```
//Manage Direct Transfer Mechanism
//

//force opposite direction on swap
LOAD_TMP_BIT 4.14       //last direction
AND_TMP_BIT 4.13        //Swapping
OR_TMP_BIT 4.9          //Run 2
SET_TMP_BIT 4.9         //Run 2

//force opposite direction on swap
LOAD_NOT_TMP_BIT 4.14   //NOT last direction
AND_TMP_BIT 4.13        //Swapping
OR_TMP_BIT 4.8          //Run 1
SET_TMP_BIT 4.8         //Run 1

//look for both directions ON
LOAD_TMP_BIT 4.8        //Run 1
AND_TMP_BIT 4.9         //Run 2
AND_BIT 683.9           //Direct Transfer Enable
AND_NOT_TMP_BIT 3.0     //NOT PLC active
SET_TMP_BIT 4.13        //save Swapping

LOAD_TMP_BIT 3.0        //PLC active
AND_TMP_BIT 4.8         //Run 1
AND_NOT_TMP_BIT 4.14    //NOT last direction
OR_TMP_BIT 4.15         //Two Wire Swap
SET_TMP_BIT 4.15        //save Two Wire Swap

LOAD_TMP_BIT 3.0        //PLC active
AND_TMP_BIT 4.9         //Run 2
AND_TMP_BIT 4.14        //last direction
OR_TMP_BIT 4.15         //Two Wire Swap
SET_TMP_BIT 4.15        //save Two Wire Swap

LOAD_TMP_BIT 3.0        //PLC active
AND_NOT_TMP_BIT 4.8     //Run 1
AND_NOT_TMP_BIT 4.9     //Run 2
OR_TMP_BIT 4.15         //Two Wire Swap
SET_TMP_BIT 4.15        //save Two Wire Swap
```

Structured Text Program (cont'd)

```
//Manage forward-reverse timer
//

LOAD_REG 541              //Forward-Reverse Time value
LOAD_NOT_TMP_BIT 51.3     //Get NOT history bit
ON_SET_TMP_REG 9, 51      //Timer Value

LOAD_NOT_TMP_BIT 4.10    //NOT Forward
AND_NOT_TMP_BIT 4.11     //NOT High Reverse
OR_TMP_BIT 4.13          //swapping
OR_TMP_BIT 11.2          //already timing
OR_TMP_BIT 4.15          //Two Wire Swap
OR_NOT_TMP_BIT 12.12     //NOT Power-up Done
SET_TMP_BIT 11, 0        //Enable Timer

TIMER_TENTHS 9,10,11     //Process forward-reverse timer

//update Swapping flags
LOAD_TMP_BIT 11.0        //Enabled
AND_TMP_BIT 11.2         //timing
SET_TMP_BIT 4.12         //Reversing Timer Active
SET_TMP_BIT 4.15         //Two Wire Swap


//
```

Structured Text Program (cont'd)

```
//Manage Forward and Reverse status bits
//

LOAD_NOT_TMP_BIT 4.12   //NOT Reversing Timer Active
OR_TMP_BIT 52.0         //Last direction-forward
AND_TMP_BIT 4.8         //Run1
AND_NOT_TMP_BIT 4.6     //NOT Stop1
AND_NOT_TMP_BIT 4.11    //NOT Reverse
SET_TMP_BIT 4.10        //save Forward
SET_TMP_BIT 52.1        //set last direction=forward

LOAD_NOT_TMP_BIT 4.12   //NOT Reversing Timer Active
OR_NOT_TMP_BIT 52.0     //NOT last direction-forward
AND_TMP_BIT 4.9         //Run2
AND_NOT_TMP_BIT 4.7     //NOT Stop2
AND_NOT_TMP_BIT 4.10    //NOT Forward

SET_TMP_BIT 4.11        //save Reverse
SET_TMP_BIT 52.2        //set last direction=reverse

LATCH 52                //last direction latch

LOAD_TMP_BIT 52.0       //Latch value (1=forward)
SET_TMP_BIT 4.14        //save Last Direction

//
// Set Outputs to IMPR
//
                        //Process Output 1
LOAD_TMP_BIT 4.10       //Forward
SET_BIT 1200.12         //Output 1
SET_BIT 1200.9          //Aux 1 LED
```

Structured Text Program (cont'd)

```
                                //Process Output 2
LOAD_TMP_BIT 4.11               //Reverse
SET_BIT 1200.13                 //Output 2
SET_BIT 1200.10                 //Aux 2 LED
SET_BIT 1200.5                  //Phase Reverse

LOAD_TMP_BIT 4.10               //Forward
OR_TMP_BIT 4.11                 //Reverse
SET_BIT 1200.0                  //Motor Run
SET_NOT_BIT 1200.1              //Motor Stop

LOAD_TMP_BIT 4.12               //Reversing Timer
SET_BIT 1200.4                  //Transition Timer

                                //Process other outputs
LOAD_BIT 455.3                  //IMPR Alarm status
SET_BIT 1200.14                 //Output 3 = Alarm
LOAD_BIT 455.2                  //IMPR Fault status
SET_NOT_BIT 1200.15             //Output 4 = Fault
LOAD_BIT 457.4                  //Reset Input LI5
SET_BIT 1200.2                  //Logic Reset
LOAD_TMP_BIT 3.0                //PLC active
SET_BIT 1200.6                  //Logic Local/Remote
LOAD_TMP_BIT 4.6                //Stop 1
OR_TMP_BIT 4.7                  //Stop 2
OR_TMP_BIT 4.12                 //Reversing Timer Active
SET_BIT 1200.11                 //Stop LED

//
// Manage Power-UP Done
//
LOAD_NOT_TMP_BIT 4.12           //Wait for power-up timer
OR_TMP_BIT 12.12                //Latch ON until next power-up
SET_TMP_BIT 12.12               //Power-up Done

// Clear PLC Control on Control Transfer
LOAD_TMP_BIT 4 0                //Control Source Transfer
AND_NOT_BIT 683 10              //NOT Bumpless
LOAD_K_REG 65532                //0xFFFC
AND_REG 704                     //mask off Run1 and Run2
ON_SET_REG 704 54               //Run bits on Bump Control Change
```

Structured Text Program for 2-Wire 2-Step Mode

Overview

The structured text program for the 2-wire 2-step mode is defined below:

Structured Text Program

```
LOGIC_ID 8   // 2-WIRE TWO STEP MODE
// Temp register allocation
// Temp 0 and Temp 1 as scratch
// Temp 2 as Requested Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
// Temp 3 as Active Control Mode
//       0=PLC
//       1=HMIpartial
//       2=TS (terminal strip)
// Temp  4 as state bits group 1
//       0=Control Transfer in process
//       1=LO1 PLC fallback value
//       2=N/A   (L02 PLC fallback value)
//       3=L01 HMI fallback value
//       4=N/A   (L02 HMI fallback value)
//       5=Global Stop
//       6=Stop1
//       7=N/A   (Stop2)
//       8=Run1
//       9=N/A   (Run2)
//      10=Step 1
//      11=Step 2
//      12=Step Timer Active
// Temp 5 as state bits group 2
//       0=Idle (Wait for Run 1)
//       1=Output 1 (Waiting for Current > 10%)
//       2=Step 1 (Waiting for Step Timer or cur. < ???)
//       3=Lockout (Waiting for transition timer 541??)
//       4=Output 2 (Waiting for stop command)
//       5=Threshold Current Detected
//
// Temp 6,7,8 as Step 1 timer
```

Structured Text Program (cont'd)

```
// Temp 9,10,11 as Lockout timer
//
// Temp 12 as INPUT History
//       1=PLC Run 1
//       2=PLC Run 2
//       3=HMI Run 1
//       4=HMI Run 2
//       5=TS Run 1
//       6=TS Run 2
//       7=Mode Change 1
//       8=
//       9=Mode Change 2
//       10=
//       11=Bumpless in Process
//       12=Power-up Done
//
// Temp 50+ as general status registers
// Temp 50  as ONSET status transition time value
// Temp 51  as ONSET status Low to High timer
// Temp 52  as ONSET status High to Low timer
// Temp 53  Latch
// Temp 54  as ONSET status  704 Run1-Run2
//
//
//Save Requested Control.in Temp 2
//
LOAD_BIT 683.8          //TS/HMI
SET_TMP_BIT 0.1         //Debounce TS/HMI in scratch
LOAD_BIT 457.5          //LI6
SET_TMP_BIT 0.0         //Debounce LI6 in scratch
SET_TMP_BIT 2.0         //PLC Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_TMP_BIT 0.1         //TS/HMI debounced
SET_TMP_BIT 2.1         //HMI Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_NOT_TMP_BIT 0.1     //TS/HMI debounced
SET_TMP_BIT 2.2         //TS Control
//
```

Structured Text Program (cont'd)

```
//Look for control transfer
//
LOAD_TMP_BIT 4.0        // Transfer in Process
SET_TMP_BIT 0.0         //save old Transfer in Process
LOAD_TMP_REG 2          //Requested Mode
COMP_TMP_REG 3, 1       //is it Active Mode
LOAD_NOT_TMP_BIT 1.2    //Not equal
SET_TMP_BIT 4.0         //Transfer in Process
//
//Manage Bump/Bumpless
//
LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_TMP_BIT 12.11   //NOT Bumpless in Process
AND_BIT 683.10          //Bumpless
SET_TMP_BIT 12.11       //Bumpless in Process (one scan)

LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_BIT 683.10      //Not bumpless
AND_NOT_TMP_BIT 0.0     //Look for Edge
SET_TMP_BIT 4.0         //Transfer in Process
SET_TMP_BIT 12.7        //Mode Change 1
SET_TMP_BIT 12.9        //Mode Change 2
//
// Save Active Control Mode in Temp Reg 3
//
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.0         //PLC requested
SET_TMP_BIT 3.0         //PLC active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.1         //HMI requested
SET_TMP_BIT 3.1         //HMI Active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.2         //TS requested
SET_TMP_BIT 3.2         //TS active
//
```

Structured Text Program (cont'd)

```
// Generate PLC Fallback Values
//
LOAD_REG 682            //PLC fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_BIT 1200.12        //last LO1 command
OR_BIT 1200.13          //last LO2 command
AND_TMP_BIT 0 2         //equal
SET_TMP_BIT 4.1         //LO1 PLC fallback
COMP_K_REG 1, 0         //---STEP(1)---
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.1          //logical or with previous value
SET_TMP_BIT 4.1         //LO1 PLC fallback
                        //---OFF(2)---- no action needed
                        //---ON(3)----- no action needed
                        //---ON OFF(4)- no action needed
                        //---OFF ON(5)- no action needed

//
// Generate HMI Fallback Values
//
LOAD_REG 645            //HMI fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_BIT 1200.12        //last LO1 command
OR_BIT 1200.13          //last LO2 command
AND_TMP_BIT 0 2         //equal
SET_TMP_BIT 4.3         //LO1 HMI fallback
COMP_K_REG 1, 0         //---STEP(1)---
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.3          //logical or with previous value
SET_TMP_BIT 4.3         //LO1 HMI fallback
                        //---OFF(2)---- no action needed
                        //---ON(3)----- no action needed
                        //---ON OFF(4)- no action needed
                        //---OFF ON(5)- no action needed

//
```

Structured Text Program (cont'd)

```
// Latch HMI Keypad info
//
LOAD_BIT 1020.12        //Aux 1
SET_TMP_BIT 13.12
LOAD_BIT 1020.13        //Aux 2
SET_TMP_BIT 13.13
LOAD_BIT 1020.14        //Stop
SET_TMP_BIT 13.14


//
// Generate Global Stop in Temp Reg 4.5
//
LOAD_TMP_BIT 3.1        //HMI Active
AND_BIT 455.2           //IMPR Fault status
OR_TMP_BIT 13.14        //HMI Stop Key
OR_BIT 456.5            //Load Shed
OR_BIT 453.1            //Diag Fault 1
OR_BIT 453.2            //Diag Fault 2
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_TMP_BIT 3.0    //NOT PLC active
AND_NOT_TMP_BIT 3.1     //NOT HMI active
AND_NOT_TMP_BIT 3.2     //NOT TS active
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_BIT 1200.0     //NOT already on
AND_NOT_BIT 1200 4      //NOT Transition Timing
AND_BIT 456.4           //Rapid Cycle
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save final Global Stop

//
//Latch comm loss values in scratch 0
//
LOAD_BIT 456.8          //PLC Comm Loss
SET_TMP_BIT 0.0         //save in scratch bit 0
LOAD_BIT 456.7          //HMI Comm Loss
SET_TMP_BIT 0.1         //save in scratch bit 1

//
```

Structured Text Program (cont'd)

```
// Generate Stop1 and Stop2 Commands
//
                          //Generate Stop1
LOAD_TMP_BIT 4.5          //Global Stop
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_TMP_BIT 0.0          //PLC Comm Loss from scratch
AND_TMP_BIT 3.0           //PLC active
AND_NOT_TMP_BIT 4.1       //NOT LO1 PLC fallback value
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_TMP_BIT 0.1          //HMI Comm Loss from scratch
AND_TMP_BIT 3.1           //HMI active
AND_NOT_TMP_BIT 4.3       //NOT LO1 HMI fallback value
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1

LOAD_TMP_BIT 3.0          //PLC active
AND_NOT_BIT 704.0         //NOT PLC Run1
AND_TMP_BIT 4.8           //Run 1
AND_NOT_TMP_BIT 0.0       //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1

LOAD_TMP_BIT 3.1          //HMI active
AND_NOT_TMP_BIT 13.12     //NOT HMI Run 1
AND_TMP_BIT 4.8           //Run 1
AND_NOT_TMP_BIT 0.1       //NOT HMI Comm Loss from scratch
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1

LOAD_TMP_BIT 3.2          //TS active
AND_NOT_BIT 457.0         //NOT TS Run 1
AND_TMP_BIT 4.8           //Run 1
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save final Stop1

                          //Generate Stop2
                          //NA
//
```

Structured Text Program (cont'd)

```
// Generate Run1 and Run2 Commands
//
                          //Generate Run 1
                          //PLC mode
LOAD_TMP_BIT 12.1         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 704.0            //PLC Network Run1
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.1          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.0           //PLC active
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

                          //HMI mode
LOAD_TMP_BIT 12.3         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
SET_TMP_BIT 12.0          //Save previous history
LOAD_TMP_BIT 13.12        //HMI Run1
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.3          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.1           //HMI active
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

                          //TS mode
LOAD_TMP_BIT 12.5         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 457.0            //LI1
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.5          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.2           //TS active
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1
```

Structured Text Program (cont'd)

```
                          //PLC Fallback
LOAD_TMP_BIT 4.1          //PLC fallback value
AND_TMP_BIT 3.0           //PLC active
AND_TMP_BIT 0.0           //PLC Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run 1

                          //HMI Fallback
LOAD_TMP_BIT 4.3          //HMI fallback value
AND_TMP_BIT 3.1           //HMI active
AND_TMP_BIT 0.1           //HMI Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run 1

                          //3wire latch
AND_NOT_TMP_BIT 4.13      //NOT Swapping
AND_NOT_TMP_BIT 12.7      //NOT Mode Change 1
SET_TMP_BIT 4.8           //save final Run 1

                          //Generate Run 2
                          //NA

//
//Manage Idle State (5.0)
//
LOAD_TMP_BIT 4.8          //Run 1
AND_NOT_TMP_BIT 4.6       //NOT Stop1
AND_TMP_BIT 5.0           //Idle State
OR_TMP_BIT 5.1            //Output 1 State
SET_TMP_BIT 5.1           //Output 1 State
LOAD_TMP_BIT 4.8          //Run 1
AND_NOT_TMP_BIT 4.6       //NOT Stop1
SET_NOT_TMP_BIT 5.0       //Idle State

// Set up Step 1 Timer
LOAD_K_BIT 1
SET_NOT_TMP_BIT 0.3       //Clear the history bit
LOAD_REG 643              //Step 1 Time value
ON_SET_TMP_REG 6,0        //set current time period
```

Structured Text Program (cont'd)

```
// Set up Lockout Timer
SET_NOT_TMP_BIT 0.3     //Clear the history bit
LOAD_REG 541            //Lockout 1 Time value
ON_SET_TMP_REG 9,0      //set current time period

//
//Manage Output 1 State (5.1)
//
//set Step 1 time period
//check for 10% FLC
LOAD_REG 466            //Average Current (%FLC)
COMP_K_REG 10, 0        //over 10% ?
LOAD_TMP_BIT 0.3        //greater than ?
AND_TMP_BIT 5.1         //Output 1 State
OR_TMP_BIT 5.2          //Step 1 State
SET_TMP_BIT 5.2         //Step 1 State

LOAD_NOT_TMP_BIT 5.0    //NOT Idle State
AND_NOT_TMP_BIT 5.2     //NOT Step 1 State
AND_NOT_TMP_BIT 5.3     //NOT Lockout State
AND_NOT_TMP_BIT 5.4     //NOT Output 2 State
SET_TMP_BIT 5.1         //Output 1 State

//
//Manage Step 1 State (5.2)
//
LOAD_TMP_BIT 5.2        //Step 1 State
SET_TMP_BIT 8.0         //enable Step 1 timer
TIMER_TENTHS 6,7,8      //process timer

//Look for current over threshold
LOAD_REG 466            //Average Current (%FLC)
COMP_REG 644, 0         //Threshold Level
LOAD_TMP_BIT 0.3        //greater than ?
OR_TMP_BIT 0.2          //equal to ?
AND_TMP_BIT 5.2         //Step 1 State
OR_TMP_BIT 5.5          //Threshold Current Detected
AND_TMP_BIT 5.2         //Step 1 State
SET_TMP_BIT 5.5         //Threshold Current Detected
```

Structured Text Program (cont'd)

```
//Look for current under threshold
LOAD_REG 466            //Average Current (%FLC)
COMP_REG 644, 0         //Threshold Level
LOAD_TMP_BIT 0.1        //less than ?
AND_TMP_BIT 5.5         //Threshold Current Detected
OR_TMP_BIT 8.1          //timed out
AND_TMP_BIT 5.2         //Step 1 State
OR_TMP_BIT 5.3          //Lockout State
SET_TMP_BIT 5.3         //Lockout State

LOAD_NOT_TMP_BIT 5.0    //NOT Idle State
AND_NOT_TMP_BIT 5.1     //NOT Output 1 State
AND_NOT_TMP_BIT 5.3     //NOT Lockout State

AND_NOT_TMP_BIT 5.4     //NOT Output 2 State
SET_TMP_BIT 5.2         //Step 1 State


//
//Manage Lockout State  (5.3)
//

LOAD_TMP_BIT 5.3        //Step 1 State
//OR_NOT_TMP_BIT 12.12    //NOT Power-up Done
SET_TMP_BIT 11.0        //enable Lockout timer
TIMER_TENTHS 9,10,11    //process timer
LOAD_TMP_BIT 11.2       //timing
SET_TMP_BIT 4.12        //Lockout timer
LOAD_TMP_BIT 11.1       //timed out
AND_TMP_BIT 5.3         //Lockout State
OR_TMP_BIT 5.4          //Step 2 State

SET_TMP_BIT 5.4         //Step 2 State

LOAD_NOT_TMP_BIT 5.0    //NOT Idle State
AND_NOT_TMP_BIT 5.1     //NOT Output 1 State
AND_NOT_TMP_BIT 5.2     //NOT Step 1 State
AND_NOT_TMP_BIT 5.4     //NOT Output 2 State
SET_TMP_BIT 5.3         //Lockout State
//
```

Structured Text Program (cont'd)

```
//Manage Output 2 State  (5.4)
//
LOAD_NOT_TMP_BIT 5.0    //NOT Idle State
AND_NOT_TMP_BIT 5.1     //NOT Output 1 State
AND_NOT_TMP_BIT 5.2     //NOT Step 1 State
AND_NOT_TMP_BIT 5.3     //NOT Lockout State
SET_TMP_BIT 5.4         //Output 2 State

//
// Set Outputs to IMPR
//
                        //Process Output 1
LOAD_TMP_BIT 5.1        //Output 1 State
OR_TMP_BIT 5.2          //Step 1 State
AND_NOT_TMP_BIT 4.6     //NOT Stop 1
SET_BIT 1200.12         //Output 1
SET_BIT 1200.9          //Aux 1 LED

                        //Process Output 2
LOAD_TMP_BIT 5.4        //Step 2 State
AND_NOT_TMP_BIT 4.6     //NOT Stop 1
SET_BIT 1200.13         //Output 2
SET_BIT 1200.10         //Aux 2 LED
SET_BIT 1200.3          //In Step 2

                        //Process Motor Run/Stop
LOAD_TMP_BIT 5.1        //Output 1 State
OR_TMP_BIT 5.2          //Step 1 State
OR_TMP_BIT 5.4          //Step 2 State
SET_BIT 1200.0          //Motor Run
SET_NOT_BIT 1200.1      //Motor Stop
SET_NOT_BIT 1200.11     //Stop LED

LOAD_TMP_BIT 4.12       //Reversing Timer
SET_BIT 1200.4          //Transition Timer
```

Structured Text Program (cont'd)

```
                                //Process other outputs
LOAD_BIT 455.3                  //IMPR Alarm status
SET_BIT 1200.14                 //Output 3 = Alarm
LOAD_BIT 455.2                  //IMPR Fault status
SET_NOT_BIT 1200.15             //Outpur 4 = Fault
LOAD_BIT 457.4                  //Reset Input LI5
SET_BIT 1200.2                  //Logic Reset
LOAD_TMP_BIT 3.0                //PLC active
SET_BIT 1200.6                  //Logic Local/Remote

//
// Manage Power-UP Done
//
LOAD_NOT_TMP_BIT 4.12           //Wait for power-up timer
OR_TMP_BIT 12.12                //Latch ON until next power-up
SET_TMP_BIT 12.12               //Power-up Done

// Clear PLC Control on Control Transfer
LOAD_TMP_BIT 4 0                //Control Source Transfer
AND_NOT_BIT 683 10              //NOT Bumpless
LOAD_K_REG 65532                //0xFFFC
AND_REG 704                     //mask off Run1 and Run2
ON_SET_REG 704 54               //Run bits on Bump Control Change
```

Structured Text Program for 3-Wire 2-Step Mode

Overview

The structured text program for the 3-wire 2-step mode is defined below:

Structured Text Program

```
LOGIC_ID 9   // 3-WIRE TWO STEP MODE
// Temp register allocation
// Temp 0 and Temp 1 as scratch
// Temp 2 as Requested Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
// Temp 3 as Active Control Mode
//       0=PLC
//       1=HMIpartial
//       2=TS (terminal strip)
// Temp  4 as state bits group 1
//       0=Control Transfer in process
//       1=LO1 PLC fallback value
//       2=N/A   (L02 PLC fallback value)
//       3=L01 HMI fallback value
//       4=N/A   (L02 HMI fallback value)
//       5=Global Stop
//       6=Stop1
//       7=N/A   (Stop2)
//       8=Run1
//       9=N/A   (Run2)
//      10=Step 1
//      11=Step 2
//      12=Step Timer Active
// Temp 5 as state bits group 2
//       0=Idle (Wait for Run 1)
//       1=Output 1 (Waiting for Current > 10%)
//       2=Step 1 (Waiting for Step Timer or cur. < ???)
//       3=Lockout (Waiting for transition timer 541??)
//       4=Output 2 (Waiting for stop command)
//       5=Threshold Current Detected
//
// Temp 6,7,8 as Step 1 timer
```

Structured Text Program (cont'd)

```
// Temp 9,10,11 as Lockout timer
//
// Temp 12 as INPUT History
//       1=PLC Run 1
//       2=PLC Run 2
//       3=HMI Run 1
//       4=HMI Run 2
//       5=TS Run 1
//       6=TS Run 2
//       7=Mode Change 1
//       8=
//       9=Mode Change 2
//       10=
//       11=Bumpless in Process
//       12=Power-up Done
//
// Temp 50+ as general status registers
// Temp 50  as ONSET status transition time value
// Temp 51  as ONSET status Low to High timer
// Temp 52  as ONSET status High to Low timer
// Temp 53  Latch
// Temp 54  as ONSET status  704 Run1-Run2
//

//
//Save Requested Control.in Temp 2
//
LOAD_BIT 683.8          //TS/HMI
SET_TMP_BIT 0.1         //Debounce TS/HMI in scratch
LOAD_BIT 457.5          //LI6
SET_TMP_BIT 0.0         //Debounce LI6 in scratch
SET_TMP_BIT 2.0         //PLC Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_TMP_BIT 0.1         //TS/HMI debounced
SET_TMP_BIT 2.1         //HMI Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_NOT_TMP_BIT 0.1     //TS/HMI debounced
SET_TMP_BIT 2.2         //TS Control
//
```

Structured Text Program (cont'd)

```
//Look for control transfer
//
LOAD_TMP_BIT 4.0        // Transfer in Process
SET_TMP_BIT 0.0         //save old Transfer in Process
LOAD_TMP_REG 2          //Requested Mode
COMP_TMP_REG 3, 1       //is it Active Mode
LOAD_NOT_TMP_BIT 1.2    //Not equal
SET_TMP_BIT 4.0         //Transfer in Process
//
//Manage Bump/Bumpless
//
LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_TMP_BIT 12.11   //NOT Bumpless in Process
SET_TMP_BIT 12.11       //Bumpless in Process (one scan)

LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_BIT 683.10      //Not bumpless
AND_NOT_TMP_BIT 0.0     //Look for Edge
SET_TMP_BIT 4.0         //Transfer in Process
SET_TMP_BIT 12.7        //Mode Change 1
SET_TMP_BIT 12.9        //Mode Change 2
//
// Save Active Control Mode in Temp Reg 3
//
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.0         //PLC requested
SET_TMP_BIT 3.0         //PLC active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.1         //HMI requested
SET_TMP_BIT 3.1         //HMI Active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.2         //TS requested
SET_TMP_BIT 3.2         //TS active
//
```

Structured Text Program (cont'd)

```
// Generate PLC Fallback Values
//
LOAD_REG 682            //PLC fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_BIT 1200.12        //last LO1 command
OR_BIT 1200.13          //last LO2 command
AND_TMP_BIT 0 2         //equal
SET_TMP_BIT 4.1         //LO1 PLC fallback
COMP_K_REG 1, 0         //---STEP(1)---
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.1          //logical or with previous value
SET_TMP_BIT 4.1         //LO1 PLC fallback
                        //---OFF(2)---- no action needed
                        //---ON(3)----- no action needed
                        //---ON OFF(4)- no action needed
                        //---OFF ON(5)- no action needed

//
// Generate HMI Fallback Values
//
LOAD_REG 645            //HMI fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_BIT 1200.12        //last LO1 command
OR_BIT 1200.13          //last LO2 command
AND_TMP_BIT 0 2         //equal
SET_TMP_BIT 4.3         //LO1 HMI fallback
COMP_K_REG 1, 0         //---STEP(1)---
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.3          //logical or with previous value
SET_TMP_BIT 4.3         //LO1 HMI fallback
                        //---OFF(2)---- no action needed
                        //---ON(3)----- no action needed
                        //---ON OFF(4)- no action needed
                        //---OFF ON(5)- no action needed

//
```

Structured Text Program (cont'd)

```
// Latch HMI Keypad info
//
LOAD_BIT 1020.12        //Aux 1
SET_TMP_BIT 13.12
LOAD_BIT 1020.13        //Aux 2
SET_TMP_BIT 13.13
LOAD_BIT 1020.14        //Stop
SET_TMP_BIT 13.14


//
// Generate Global Stop in Temp Reg 4.5
//
LOAD_TMP_BIT 13.14      //HMI Stop Key
OR_NOT_BIT 457.3        //NOT LI 4
OR_BIT 456.5            //Load Shed
OR_BIT 453.1            //Diag Fault 1
OR_BIT 453.2            //Diag Fault 2
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_TMP_BIT 3.0    //NOT PLC active
AND_NOT_TMP_BIT 3.1     //NOT HMI active
AND_NOT_TMP_BIT 3.2     //NOT TS active
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_BIT 1200.0     //NOT already on
AND_NOT_BIT 1200 4      //NOT Transition Timing
AND_BIT 456.4           //Rapid Cycle
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save final Global Stop

//
//Latch comm loss values in scratch 0
//
LOAD_BIT 456.8          //PLC Comm Loss
SET_TMP_BIT 0.0         //save in scratch bit 0
LOAD_BIT 456.7          //HMI Comm Loss
SET_TMP_BIT 0.1         //save in scratch bit 1

//
```

Structured Text Program (cont'd)

```
// Generate Stop1 and Stop2 Commands
//
                          //Generate Stop1
LOAD_TMP_BIT 4.5          //Global Stop
OR_NOT_TMP_BIT 12.12      //NOT Powerup Done
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_TMP_BIT 0.0          //PLC Comm Loss from scratch
AND_TMP_BIT 3.0           //PLC active
AND_NOT_TMP_BIT 4.1       //NOT LO1 PLC fallback value
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_TMP_BIT 0.1          //HMI Comm Loss from scratch
AND_TMP_BIT 3.1           //HMI active
AND_NOT_TMP_BIT 4.3       //NOT LO1 HMI fallback value
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1

LOAD_TMP_BIT 3.0          //PLC active
AND_NOT_BIT 704.0         //NOT PLC Run1
AND_TMP_BIT 4.8           //Run 1
AND_NOT_TMP_BIT 0.0       //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save final Stop1

                          //Generate Stop2
                          //NA
//
```

Structured Text Program (cont'd)

```
// Generate Run1 and Run2 Commands
//
                          //Generate Run 1
                          //PLC mode
LOAD_TMP_BIT 12.1         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 704.0            //PLC Network Run1
AND_TMP_BIT 12.12         //Power-up Done
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
SET_TMP_BIT 12.1          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.0           //PLC active
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
AND_NOT_TMP_BIT 0.0       //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

                          //HMI mode
LOAD_TMP_BIT 12.3         //Input history
SET_TMP_BIT 12.0          //Save previous history
LOAD_TMP_BIT 13.12        //HMI Run1
SET_TMP_BIT 12.3          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.1           //HMI active
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
AND_NOT_TMP_BIT 0.1       //NOT HMI Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

                          //TS mode
LOAD_TMP_BIT 12.5         //Input history
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 457.0            //LI1
SET_TMP_BIT 12.5          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.2           //TS active
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1
```

Structured Text Program (cont'd)

```
                              //PLC Fallback
LOAD_TMP_BIT 4.1              //PLC fallback value
AND_TMP_BIT 3.0               //PLC active
AND_TMP_BIT 0.0               //PLC Comm Loss from scratch
OR_TMP_BIT 4.8                //Include previous result
SET_TMP_BIT 4.8               //save partial Run 1

                              //HMI Fallback
LOAD_TMP_BIT 4.3              //HMI fallback value
AND_TMP_BIT 3.1               //HMI active
AND_TMP_BIT 0.1               //HMI Comm Loss from scratch
OR_TMP_BIT 4.8                //Include previous result
SET_TMP_BIT 4.8               //save partial Run 1

                              //3wire latch
AND_NOT_TMP_BIT 4.6           //NOT Stop 1
AND_NOT_TMP_BIT 4.13          //NOT Swapping
AND_NOT_TMP_BIT 12.7          //NOT Mode Change 1
SET_TMP_BIT 4.8               //save final Run 1

                              //Generate Run 2
                              //NA

//
//Manage Idle State (5.0)
//
LOAD_TMP_BIT 4.8              //Run 1
AND_TMP_BIT 5.0               //Idle State
OR_TMP_BIT 5.1                //Output 1 State
SET_TMP_BIT 5.1               //Output 1 State
LOAD_TMP_BIT 4.8              //Run 1
SET_NOT_TMP_BIT 5.0           //Idle State

// Set up Step 1 Timer
LOAD_K_BIT 1
SET_NOT_TMP_BIT 0.3           //Clear the history bit
LOAD_REG 643                  //Step 1 Time value
ON_SET_TMP_REG 6,0            //set current time period
```

Structured Text Program (cont'd)

```
// Set up Lockout Timer
SET_NOT_TMP_BIT 0.3     //Clear the history bit
LOAD_REG 541            //Lockout 1 Time value
ON_SET_TMP_REG 9,0      //set current time period

//
//Manage Output 1 State (5.1)
//
//set Step 1 time period
//check for 10% FLC
LOAD_REG 466            //Average Current (%FLC)
COMP_K_REG 10, 0        //over 10% ?
LOAD_TMP_BIT 0.3        //greater than ?
AND_TMP_BIT 5.1         //Output 1 State
OR_TMP_BIT 5.2          //Step 1 State
SET_TMP_BIT 5.2         //Step 1 State

LOAD_NOT_TMP_BIT 5.0    //NOT Idle State
AND_NOT_TMP_BIT 5.2     //NOT Step 1 State
AND_NOT_TMP_BIT 5.3     //NOT Lockout State
AND_NOT_TMP_BIT 5.4     //NOT Output 2 State
SET_TMP_BIT 5.1         //Output 1 State

//
//Manage Step 1 State (5.2)
//
LOAD_TMP_BIT 5.2        //Step 1 State
//OR_NOT_TMP_BIT 12.12    //NOT Power-up Done
SET_TMP_BIT 8.0         //enable Step 1 timer
TIMER_TENTHS 6,7,8      //process timer

//Look for current over threshold
LOAD_REG 466            //Average Current (%FLC)
COMP_REG 644, 0         //Threshold Level
LOAD_TMP_BIT 0.3        //greater than ?
OR_TMP_BIT 0.2          //equal to ?
AND_TMP_BIT 5.2         //Step 1 State
OR_TMP_BIT 5.5          //Threshold Current Detected
AND_TMP_BIT 5.2         //Step 1 State
SET_TMP_BIT 5.5         //Threshold Current Detected
```

Structured Text Program (cont'd)

```
//Look for current under threshold
LOAD_REG 466            //Average Current (%FLC)
COMP_REG 644, 0         //Threshold Level
LOAD_TMP_BIT 0.1        //less than ?
AND_TMP_BIT 5.5         //Threshold Current Detected
OR_TMP_BIT 8.1          //timed out
AND_TMP_BIT 5.2         //Step 1 State
OR_TMP_BIT 5.3          //Lockout State
SET_TMP_BIT 5.3         //Lockout State

LOAD_NOT_TMP_BIT 5.0    //NOT Idle State
AND_NOT_TMP_BIT 5.1     //NOT Output 1 State
AND_NOT_TMP_BIT 5.3     //NOT Lockout State
AND_NOT_TMP_BIT 5.4     //NOT Output 2 State
SET_TMP_BIT 5.2         //Step 1 State


//
//Manage Lockout State  (5.3)
//

LOAD_TMP_BIT 5.3        //Step 1 State
SET_TMP_BIT 11.0        //enable Lockout timer
TIMER_TENTHS 9,10,11    //process timer
LOAD_TMP_BIT 11.2       //timing
SET_TMP_BIT 4.12        //Lockout timer
LOAD_TMP_BIT 11.1       //timed out
AND_TMP_BIT 5.3         //Lockout State
OR_TMP_BIT 5.4          //Step 2 State

SET_TMP_BIT 5.4         //Step 2 State

LOAD_NOT_TMP_BIT 5.0    //NOT Idle State
AND_NOT_TMP_BIT 5.1     //NOT Output 1 State
AND_NOT_TMP_BIT 5.2     //NOT Step 1 State
AND_NOT_TMP_BIT 5.4     //NOT Output 2 State
SET_TMP_BIT 5.3         //Lockout State


//
```

Structured Text Program (cont'd)

```
//Manage Output 2 State  (5.4)
//
LOAD_NOT_TMP_BIT 5.0    //NOT Idle State
AND_NOT_TMP_BIT 5.1     //NOT Output 1 State
AND_NOT_TMP_BIT 5.2     //NOT Step 1 State
AND_NOT_TMP_BIT 5.3     //NOT Lockout State
SET_TMP_BIT 5.4         //Output 2 State

//
// Set Outputs to IMPR
//
                        //Process Output 1
LOAD_TMP_BIT 5.1        //Output 1 State
OR_TMP_BIT 5.2          //Step 1 State
AND_NOT_TMP_BIT 4.6     //NOT Stop 1
SET_BIT 1200.12         //Output 1
SET_BIT 1200.9          //Aux 1 LED

                        //Process Output 2
LOAD_TMP_BIT 5.4        //Step 2 State
AND_NOT_TMP_BIT 4.6     //NOT Stop 1
SET_BIT 1200.13         //Output 2
SET_BIT 1200.10         //Aux 2 LED
SET_BIT 1200.3          //In Step 2

                        //Process Motor Run/Stop
LOAD_TMP_BIT 5.1        //Output 1 State
OR_TMP_BIT 5.2          //Step 1 State
OR_TMP_BIT 5.4          //Step 2 State
SET_BIT 1200.0          //Motor Run
SET_NOT_BIT 1200.1      //Motor Stop
SET_NOT_BIT 1200.11     //Stop LED

LOAD_TMP_BIT 4.12       //Reversing Timer
SET_BIT 1200.4          //Transition Timer
```

Structured Text Program (cont'd)

```
                                //Process other outputs
LOAD_BIT 455.3                  //IMPR Alarm status
SET_BIT 1200.14                 //Output 3 = Alarm
LOAD_BIT 455.2                  //IMPR Fault status
SET_NOT_BIT 1200.15             //Outpur 4 = Fault
LOAD_BIT 457.4                  //Reset Input LI5
SET_BIT 1200.2                  //Logic Reset
LOAD_TMP_BIT 3.0                //PLC active
SET_BIT 1200.6                  //Logic Local/Remote

//
// Manage Power-UP Done
//
LOAD_NOT_TMP_BIT 4.12           //Wait for power-up timer
OR_TMP_BIT 12.12                //Latch ON until next power-up
SET_TMP_BIT 12.12               //Power-up Done

// Clear PLC Control on Control Transfer
LOAD_TMP_BIT 4 0                //Control Source Transfer
AND_NOT_BIT 683 10              //NOT Bumpless
LOAD_K_REG 65532                //0xFFFC
AND_REG 704                     //mask off Run1 and Run2
ON_SET_REG 704 54               //Run bits on Bump Control Change
```

Structured Text Program for 2-Wire 2-Speed Mode

Overview

The structured text program for the 2-wire 2-speed mode is defined below:

Structured Text Program

```
LOGIC_ID 10  //2-WIRE TWO SPEED MODE
// Temp register allocation
// Temp 0 and Temp 1 as scratch
// Temp 2 as Requested Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
// Temp 3 as Active Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
// Temp  4 as state bits group 1
//       0=Control Transfer in process
//       1=LO1 PLC fallback value
//       2=L02 PLC fallback value
//       3=L01 HMI fallback value
//       4=L02 HMI fallback value
//       5=Global Stop
//       6=Stop1
//       7=Stop2
//       8=Run1
//       9=Run2
//       10=Low Speed
//       11=High Speed
//       12=Lockout Timer Active
//
// Temp 5 as state bits group 2
//
// Temp 9, 10, 11 as Speed change timer
//
// Temp 12 as INPUT History
//       1=PLC Run 1
//       2=PLC Run 2
//       3=HMI Run 1
```

Structured Text Program (cont'd)

```
//       4=HMI Run 2
//       5=TS Run 1
//       6=TS Run 2
//       7=Mode Change 1
//       8=
//       9=Mode Change 2
//       10=
//       11=Bumpless in Process
//       12=Power-up Done
//
// Temp 50+ as general status registers
// Temp 50  as ONSET status transition time value
// Temp 51  as ONSET status Low to High timer
// Temp 52  as ONSET status High to Low timer
// Temp 53  Latch
// Temp 54  as ONSET status  704 Run1-Run2
//
//Save Requested Control.in Temp 2
//
LOAD_BIT 683.8          //TS/HMI
SET_TMP_BIT 0.1         //Debounce TS/HMI in scratch
LOAD_BIT 457.5          //LI6
SET_TMP_BIT 0.0         //Debounce LI6 in scratch
SET_TMP_BIT 2.0         //PLC Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_TMP_BIT 0.1         //TS/HMI debounced
SET_TMP_BIT 2.1         //HMI Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_NOT_TMP_BIT 0.1     //TS/HMI debounced
SET_TMP_BIT 2.2         //TS Control
//
//Look for control transfer
//
LOAD_TMP_BIT 4.0        // Transfer in Process
SET_TMP_BIT 0.0         //save old Transfer in Process
LOAD_TMP_REG 2          //Requested Mode
COMP_TMP_REG 3, 1       //is it Active Mode
LOAD_NOT_TMP_BIT 1.2    //Not equal
SET_TMP_BIT 4.0         //Transfer in Process
//
```

Structured Text Program (cont'd)

```
//Manage Bump/Bumpless
//
LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_TMP_BIT 12.11   //NOT Bumpless in Process
AND_BIT 683.10          //Bumpless  ****new !!!
SET_TMP_BIT 12.11       //Bumpless in Process (one scan)

LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_BIT 683.10      //Not bumpless
AND_NOT_TMP_BIT 0.0     //Look for Edge
SET_TMP_BIT 4.0         //Transfer in Process
SET_TMP_BIT 12.7        //Mode Change 1
SET_TMP_BIT 12.9        //Mode Change 2

//
// Save Active Control Mode in Temp Reg 3
//
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.0         //PLC requested
SET_TMP_BIT 3.0         //PLC active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.1         //HMI requested
SET_TMP_BIT 3.1         //HMI Active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.2         //TS requested
SET_TMP_BIT 3.2         //TS active
//
```

Structured Text Program (cont'd)

```
// Generate PLC Fallback Values
//
LOAD_REG 682            //PLC fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.12         //last LO1 command
SET_TMP_BIT 4.1         //LO1 PLC fallback
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.13         //last LO2 command
SET_TMP_BIT 4.2         //LO2 PLC fallback
                        //---STEP(1)--- no action needed
                        //---OFF(2)---- no action needed
                        //---ON(3)----- no action needed
COMP_K_REG 4, 0         //---ON OFF(4)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.1          //logical or with previous value
SET_TMP_BIT 4.1         //LO1 PLC fallback
COMP_K_REG 5, 0         //---OFF ON(5)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.2          //logical or with previous value
SET_TMP_BIT 4.2         //LO2 PLC fallback

//
```

Structured Text Program (cont'd)

```
// Generate HMI Fallback Values
//
LOAD_REG 645            //HMI fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.12         //last LO1 command
SET_TMP_BIT 4.3         //LO1 HMI fallback
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.13         //last LO2 command
SET_TMP_BIT 4.4         //LO2 HMI fallback
                        //---STEP(1)--- no action needed
                        //---OFF(2)---- no action needed
                        //---ON(3)----- no action needed
COMP_K_REG 4, 0         //---ON OFF(4)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.3          //logical or with previous value
SET_TMP_BIT 4.3         //LO1 HMI fallback
COMP_K_REG 5, 0         //---OFF ON(5)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.4          //logical or with previous value
SET_TMP_BIT 4.4         //LO2 HMI fallback

//
// Latch HMI Keypad info
//
LOAD_BIT 1020.12        //Aux 1
SET_TMP_BIT 13.12
LOAD_BIT 1020.13        //Aux 2
SET_TMP_BIT 13.13
LOAD_BIT 1020.14        //Stop
SET_TMP_BIT 13.14

//
```

Structured Text Program (cont'd)

```
// Generate Global Stop in Temp Reg 4.5
//
LOAD_TMP_BIT 3.1        //HMI Active
AND_BIT 455.2           //IMPR Fault status
OR_TMP_BIT 13.14        //HMI Stop Key
OR_BIT 456.5            //Load Shed
OR_BIT 453.1            //Diag Fault 1
OR_BIT 453.2            //Diag Fault 2
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_TMP_BIT 3.0    //NOT PLC active
AND_NOT_TMP_BIT 3.1     //NOT HMI active
AND_NOT_TMP_BIT 3.2     //NOT TS active
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_BIT 1200.0     //NOT already on
AND_BIT 456.4           //Rapid Cycle
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save final Global Stop

//
//Latch comm loss values in scratch 0
//
LOAD_BIT 456.8         //PLC Comm Loss
SET_TMP_BIT 0.0        //save in scratch bit 0
LOAD_BIT 456.7         //HMI Comm Loss
SET_TMP_BIT 0.1        //save in scratch bit 1

//
```

Structured Text Program (cont'd)

```
// Generate Stop1 and Stop2 Commands
//
                          //Generate Stop1
LOAD_TMP_BIT 4.5          //Global Stop
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_TMP_BIT 0.0          //PLC Comm Loss from scratch
AND_TMP_BIT 3.0           //PLC active
AND_NOT_TMP_BIT 4.1       //NOT LO1 PLC fallback value
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_TMP_BIT 0.1          //HMI Comm Loss from scratch
AND_TMP_BIT 3.1           //HMI active
AND_NOT_TMP_BIT 4.3       //NOT LO1 HMI fallback value
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1

LOAD_NOT_BIT 704.0        //NOT PLC Run1
OR_NOT_BIT 704.6          //NOT Low Speed
AND_TMP_BIT 3.0           //PLC active
AND_TMP_BIT 4.8           //Run 1
AND_NOT_TMP_BIT 0.0       //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1

LOAD_TMP_BIT 3.1          //HMI active
AND_NOT_TMP_BIT 13.12     //NOT HMI Run 1
AND_TMP_BIT 4.8           //Run 1
AND_NOT_TMP_BIT 0.1       //NOT HMI Comm Loss from scratch
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1

LOAD_TMP_BIT 3.2          //TS active
AND_NOT_BIT 457.0         //NOT TS Run 1
AND_TMP_BIT 4.8           //Run 1
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save final Stop1
```

Structured Text Program (cont'd)

```
                                //Generate Stop2
LOAD_TMP_BIT 4.5                //Global Stop
SET_TMP_BIT 4.7                 //save partial Stop2
LOAD_TMP_BIT 0.0                //PLC Comm Loss from scratch
AND_TMP_BIT 3.0                 //PLC active
AND_NOT_TMP_BIT 4.2             //NOT LO2 PLC fallback value
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2
LOAD_TMP_BIT 0.1                //HMI Comm Loss from scratch
AND_TMP_BIT 3.1                 //HMI active
AND_NOT_TMP_BIT 4.4             //NOT LO1 HMI fallback value
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2

LOAD_NOT_BIT 704.0              //NOT PLC Run2
OR_BIT 704.6                    //Low Speed
AND_TMP_BIT 3.0                 //PLC active
AND_TMP_BIT 4.9                 //Run 2
AND_NOT_TMP_BIT 0.0             //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2

LOAD_TMP_BIT 3.1                //HMI active
AND_NOT_TMP_BIT 13.13           //NOT HMI Run 2
AND_TMP_BIT 4.9                 //Run 2
AND_NOT_TMP_BIT 0.1             //NOT HMI Comm Loss from scratch
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2

LOAD_TMP_BIT 3.2                //TS active
AND_NOT_BIT 457.1               //NOT TS Run 2
AND_TMP_BIT 4.9                 //Run 2
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save final Stop2

//
```

Structured Text Program (cont'd)

```
// Generate Run1 and Run2 Commands
//
                          //Generate Run 1
                          //PLC mode
LOAD_TMP_BIT 12.1         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
AND_NOT_TMP_BIT 4.6       //NOT Stop1
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 704.0            //PLC Network Run1
AND_TMP_BIT 12.12         //Power-up Done
AND_BIT 704.6             //Low Speed
SET_TMP_BIT 12.1          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.0           //PLC active
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

                         //HMI mode
LOAD_TMP_BIT 12.3        //Input history
AND_NOT_TMP_BIT 12.11    //NOT Bumpless in Process
AND_NOT_TMP_BIT 4.6      //NOT Stop1
SET_TMP_BIT 12.0         //Save previous history
LOAD_TMP_BIT 13.12       //HMI Run1
AND_TMP_BIT 12.12        //Power-up Done
SET_TMP_BIT 12.3         //Save new history
AND_NOT_TMP_BIT 12.0     //NOT previous history
AND_TMP_BIT 3.1          //HMI active
OR_TMP_BIT 4.8           //Include previous result
SET_TMP_BIT 4.8          //save partial Run1
```

Structured Text Program (cont'd)

```
                          //TS mode
LOAD_TMP_BIT 12.5         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
AND_NOT_TMP_BIT 4.6       //NOT Stop1
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 457.0            //TS Run1
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.5          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.2           //TS active
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

                          //PLC Fallback
LOAD_TMP_BIT 4.1          //PLC fallback value
AND_TMP_BIT 3.0           //PLC active
AND_TMP_BIT 0.0           //PLC Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run 1

                          //HMI Fallback
LOAD_TMP_BIT 4.3          //HMI fallback value
AND_TMP_BIT 3.1           //HMI active
AND_TMP_BIT 0.1           //HMI Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run 1

                          //3wire latch
AND_NOT_TMP_BIT 12.7      //NOT Mode Change 1
SET_TMP_BIT 4.8           //save final Run 1
```

Structured Text Program (cont'd)

```
                          //Generate Run 2
                          //PLC mode
LOAD_TMP_BIT 12.2         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
AND_NOT_TMP_BIT 4.7       //NOT Stop2
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 704.0            //PLC Network Run2
AND_TMP_BIT 12.12         //Power-up Done
AND_NOT_BIT 704.6         //NOT Low Speed
SET_TMP_BIT 12.2          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.0           //PLC active
OR_TMP_BIT 4.9            //Include previous result
SET_TMP_BIT 4.9           //save partial Run2

                          //HMI mode
LOAD_TMP_BIT 12.4         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
AND_NOT_TMP_BIT 4.7       //NOT Stop2
SET_TMP_BIT 12.0          //Save previous history
LOAD_TMP_BIT 13.13        //HMI Run2
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.4          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.1           //HMI active
OR_TMP_BIT 4.9            //Include previous result
SET_TMP_BIT 4.9           //save partial Run2

                          //TS mode
LOAD_TMP_BIT 12.6         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
AND_NOT_TMP_BIT 4.7       //NOT Stop2
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 457.1            //TS Run2
AND_TMP_BIT 12.12         //Power-up Done
SET_TMP_BIT 12.6          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.2           //TS active
OR_TMP_BIT 4.9            //Include previous result
SET_TMP_BIT 4.9           //save partial Run2
```

Structured Text Program (cont'd)

```
                                //PLC Fallback
LOAD_TMP_BIT 4.2                //PLC fallback value
AND_TMP_BIT 3.0                 //PLC active
AND_TMP_BIT 0.0                 //PLC Comm Loss from scratch
OR_TMP_BIT 4.9                  //Include previous result
SET_TMP_BIT 4.9                 //save partial Run2

                                //HMI Fallback
LOAD_TMP_BIT 4.4                //HMI fallback value
AND_TMP_BIT 3.1                 //HMI active
AND_TMP_BIT 0.1                 //HMI Comm Loss from scratch
OR_TMP_BIT 4.9                  //Include previous result
SET_TMP_BIT 4.9                 //save partial Run2

                                //3wire latch
AND_NOT_TMP_BIT 12.9            //NOT Mode Change 2
SET_TMP_BIT 4.9                 //save final Run 2

//
```

Structured Text Program (cont'd)

```
//Manage Speed Change timer
//

LOAD_K_REG 1             //Low to High Time value (.1s)
LOAD_TMP_BIT 4.10        //Low Speed
ON_SET_TMP_REG 9, 51     //Timer Value - Low to High
LOAD_REG 541             //High to Low Time value
LOAD_TMP_BIT 4.11        //High Speed
OR_NOT_TMP_BIT 12.12     //NOT Power-up Done
ON_SET_TMP_REG 9, 52     //Timer Value - High to Low

LOAD_NOT_TMP_BIT 4.8     //NOT Run1
OR_TMP_BIT 4.6           //Stop1
AND_TMP_BIT 4.10         //Low Speed
SET_TMP_BIT 0.0          //save partial result in scratch

LOAD_NOT_TMP_BIT 4.9     //NOT Run2
OR_TMP_BIT 4.7           //Stop2
AND_TMP_BIT 4.11         //High Speed
OR_TMP_BIT 0.0           //include partial result
SET_TMP_BIT 0.0          //save partial result in scratch

LOAD_NOT_TMP_BIT 4.10    //NOT Low Speed
AND_NOT_TMP_BIT 4.11     //NOT High Speed
AND_TMP_BIT 11.2         //already timing
OR_TMP_BIT 0.0           //include partial result
OR_NOT_TMP_BIT 12.12     //NOT Power-up Done
SET_TMP_BIT 11.0         //Enable Timer

TIMER_TENTHS  9,10,11    //Process Speed Change timer

//
```

Structured Text Program (cont'd)

```
//update Lockout Timing flag
//

LOAD_TMP_BIT 11.0       //Enabled
AND_TMP_BIT 11.2        //timing
SET_TMP_BIT 4.12        //Reversing Timer Active

//
//Manage Low Speed and High Speed status bits
//

LOAD_NOT_TMP_BIT 11.2  //NOT timing status
OR_TMP_BIT 53.0        //last Speed
AND_TMP_BIT 4.8        //Run1
AND_NOT_TMP_BIT 4.6    //NOT Stop1
AND_NOT_TMP_BIT 4.11   //NOT High Speed
SET_TMP_BIT 4.10       //save Low Speed
SET_TMP_BIT 53.1       //set last Speed

LOAD_NOT_TMP_BIT 11.2  //NOT timing status
OR_NOT_TMP_BIT 53.0    //NOT last Speed
AND_TMP_BIT 4.9        //Run2
AND_NOT_TMP_BIT 4.7    //NOT Stop2
AND_NOT_TMP_BIT 4.10   //NOT Low Speed
SET_TMP_BIT 4.11       //save High Speed
SET_TMP_BIT 53.2       //set last Speed

LATCH 53               //last Speed latch

//
// Set Outputs to IMPR
//
                       //Process Output 1
LOAD_TMP_BIT 4.10      //Low Speed
SET_BIT 1200.12        //Output 1
SET_BIT 1200.9         //Aux 1 LED
```

Structured Text Program (cont'd)

```
                          //Process Output 2
LOAD_TMP_BIT 4.11         //High Speed
SET_BIT 1200.13           //Output 2
SET_BIT 1200.10           //Aux 2 LED
SET_BIT 1200.7            //select FLA 2

LOAD_TMP_BIT 4.10         //Low Speed
OR_TMP_BIT 4.11           //High Speed
SET_BIT 1200.0            //Motor Run
SET_NOT_BIT 1200.1        //Motor Stop

LOAD_TMP_BIT 4.12          //Reversing Timer
SET_BIT 1200.4             //Transition Timer

                          //Process other outputs
LOAD_BIT 455.3            //IMPR Alarm status
SET_BIT 1200.14           //Output 3 = Alarm
LOAD_BIT 455.2            //IMPR Fault status
SET_NOT_BIT 1200.15       //Output 4 = Fault
LOAD_BIT 457.4            //Reset Input LI5
SET_BIT 1200.2            //Logic Reset
LOAD_TMP_BIT 3.0          //PLC active
SET_BIT 1200.6            //Logic Local/Remote
LOAD_TMP_BIT 4.5          //Global Stop
OR_TMP_BIT 11.2           //timing
SET_BIT 1200.11           //Stop LED

//
// Manage Power-UP Done
//
LOAD_NOT_TMP_BIT 4.12   //Wait for power-up timer
OR_TMP_BIT 12.12          //Latch ON until next power-up
SET_TMP_BIT 12.12         //Power-up Done

// Clear PLC Control on Control Transfer
LOAD_TMP_BIT 4 0          //Control Source Transfer
AND_NOT_BIT 683 10        //NOT Bumpless
LOAD_K_REG 65532          //0xFFFC
AND_REG 704               //mask off Run1 and Run2
ON_SET_REG 704 54         //Run bits on Bump Control Change
```

Structured Text Program for 3-Wire 2-Speed Mode

Overview

The structured text program for the 3-wire 2-speed mode is defined below:

Structured Text Program

```
LOGIC_ID 11    //  3-WIRE TWO SPEED MODE
// Temp register allocation
// Temp 0 and Temp 1 as scratch
// Temp 2 as Requested Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
//
// Temp 3 as Active Control Mode
//       0=PLC
//       1=HMI
//       2=TS (terminal strip)
//
// Temp  4 as state bits group 1
//       0=Control Transfer in process
//       1=LO1 PLC fallback value
//       2=L02 PLC fallback value
//       3=L01 HMI fallback value
//       4=L02 HMI fallback value
//       5=Global Stop
//       6=Stop1
//       7=Stop2
//       8=Run1
//       9=Run2
//      10=Speed 1
//      11=Speed 2
//      12=Lockout Timer Active
//      13=Swapping
//      14=Last Speed
//      15=Two Wire Swap
//
// Temp 5 as state bits group 2
//
// Temp 9, 10, 11 as Lockout Timer
```

Structured Text Program (cont'd)

```
// Temp 12 as INPUT History
//       1=PLC Run 1
//       2=PLC Run 2
//       3=HMI Run 1
//       4=HMI Run 2
//       5=TS Run 1
//       6=TS Run 2
//       7=Mode Change 1
//       8=
//       9=Mode Change 2
//       10=
//       11=Bumpless in Process
//       12=Power-up Done
//
// Temp 50+ as general status registers
// Temp 50  as ONSET status transition time value
// Temp 51  as ONSET status Low to High timer
// Temp 52  as ONSET status High to Low timer
// Temp 53  Latch
// Temp 54  as ONSET status  704 Run1-Run2
//
//Save Requested Control.in Temp 2
//
LOAD_BIT 683.8          //TS/HMI
SET_TMP_BIT 0.1         //Debounce TS/HMI in scratch
LOAD_BIT 457.5          //LI6
SET_TMP_BIT 0.0         //Debounce LI6 in scratch
SET_TMP_BIT 2.0         //PLC Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_TMP_BIT 0.1         //TS/HMI debounced
SET_TMP_BIT 2.1         //HMI Control
LOAD_NOT_TMP_BIT 0.0    //LI6 debounced
AND_NOT_TMP_BIT 0.1     //TS/HMI debounced
SET_TMP_BIT 2.2         //TS Control
//
```

Structured Text Program (cont'd)

```
//Look for control transfer
//
LOAD_TMP_BIT 4.0        // Transfer in Process
SET_TMP_BIT 0.0         //save old Transfer in Process
LOAD_TMP_REG 2          //Requested Mode
COMP_TMP_REG 3, 1       //is it Active Mode
LOAD_NOT_TMP_BIT 1.2    //Not equal
SET_TMP_BIT 4.0         //Transfer in Process
//
//Manage Bump/Bumpless
//
LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_TMP_BIT 12.11   //NOT Bumpless in Process
SET_TMP_BIT 12.11       //Bumpless in Process (one scan)

LOAD_TMP_BIT 4.0        //Transfer in Process
AND_NOT_BIT 683.10      //Not bumpless
AND_NOT_TMP_BIT 0.0     //Look for Edge
SET_TMP_BIT 4.0         //Transfer in Process
SET_TMP_BIT 12.7        //Mode Change 1
SET_TMP_BIT 12.9        //Mode Change 2

//
// Save Active Control Mode in Temp Reg 3
//
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.0         //PLC requested
SET_TMP_BIT 3.0         //PLC active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.1         //HMI requested
SET_TMP_BIT 3.1         //HMI Active
LOAD_NOT_TMP_BIT 4, 0   //not Transfer in Process
AND_TMP_BIT 2.2         //TS requested
SET_TMP_BIT 3.2         //TS active
//
```

Structured Text Program (cont'd)

```
// Generate PLC Fallback Values
//
LOAD_REG 682            //PLC fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.12         //last LO1 command
SET_TMP_BIT 4.1         //LO1 PLC fallback
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.13         //last LO2 command
SET_TMP_BIT 4.2         //LO2 PLC fallback
                        //---STEP(1)--- no action needed
                        //---OFF(2)---- no action needed
                        //---ON(3)----- no action needed
COMP_K_REG 4, 0         //---ON OFF(4)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.1          //logical or with previous value
SET_TMP_BIT 4.1         //LO1 PLC fallback
COMP_K_REG 5, 0         //---OFF ON(5)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.2          //logical or with previous value
SET_TMP_BIT 4.2         //LO2 PLC fallback

//
```

Structured Text Program (cont'd)

```
// Generate HMI Fallback Values
//
LOAD_REG 645            //HMI fallback mode
COMP_K_REG 0, 0         //---HOLD(0)---
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.12         //last LO1 command
SET_TMP_BIT 4.3         //LO1 HMI fallback
LOAD_TMP_BIT 0 2        //equal
AND_BIT 1200.13         //last LO2 command
SET_TMP_BIT 4.4         //LO2 HMI fallback
                        //---STEP(1)--- no action needed
                        //---OFF(2)---- no action needed
                        //---ON(3)----- no action needed
COMP_K_REG 4, 0         //---ON OFF(4)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.3          //logical or with previous value
SET_TMP_BIT 4.3         //LO1 HMI fallback
COMP_K_REG 5, 0         //---OFF ON(5)----
LOAD_K_BIT 1            //fallback to ON
AND_TMP_BIT 0 2         //equal
OR_TMP_BIT 4.4          //logical or with previous value
SET_TMP_BIT 4.4         //LO2 HMI fallback

//
// Latch HMI Keypad info
//
LOAD_BIT 1020.12        //Aux 1
SET_TMP_BIT 13.12
LOAD_BIT 1020.13        //Aux 2
SET_TMP_BIT 13.13
LOAD_BIT 1020.14        //Stop
SET_TMP_BIT 13.14

//
```

Structured Text Program (cont'd)

```
// Generate Global Stop in Temp Reg 4.5
//
LOAD_TMP_BIT 13.14      //HMI Stop Key
OR_NOT_BIT 457.3        //NOT Stop
OR_BIT 456.5            //Load Shed
OR_BIT 453.1            //Diag Fault 1
OR_BIT 453.2            //Diag Fault 2
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_TMP_BIT 3.0    //NOT PLC active
AND_NOT_TMP_BIT 3.1     //NOT HMI active
AND_NOT_TMP_BIT 3.2     //NOT TS active
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save partial Global Stop
LOAD_NOT_BIT 1200.0     //NOT already on
AND_BIT 456.4           //Rapid Cycle
OR_TMP_BIT 4.5          //include partial Global Stop
SET_TMP_BIT 4.5         //Save final Global Stop

//
//Latch comm loss values in scratch 0
//
LOAD_BIT 456.8          //PLC Comm Loss
SET_TMP_BIT 0.0         //save in scratch bit 0
LOAD_BIT 456.7          //HMI Comm Loss
SET_TMP_BIT 0.1         //save in scratch bit 1

//
```

Structured Text Program (cont'd)

```
// Generate Stop1 and Stop2 Commands
//
                          //Generate Stop1
LOAD_TMP_BIT 4.5          //Global Stop
OR_NOT_TMP_BIT 12.12      //NOT Powerup Done
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_TMP_BIT 0.0          //PLC Comm Loss from scratch
AND_TMP_BIT 3.0           //PLC active
AND_NOT_TMP_BIT 4.1       //NOT LO1 PLC fallback value
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1
LOAD_TMP_BIT 0.1          //HMI Comm Loss from scratch
AND_TMP_BIT 3.1           //HMI active
AND_NOT_TMP_BIT 4.3       //NOT LO1 HMI fallback value
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save partial Stop1

LOAD_NOT_BIT 704.0        //NOT PLC Run1
OR_NOT_BIT 704.6          //NOT Low Speed
AND_TMP_BIT 3.0           //PLC active
AND_TMP_BIT 4.8           //Run 1
AND_NOT_TMP_BIT 0.0       //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.6            //Include partial Stop1
SET_TMP_BIT 4.6           //save final Stop1
```

Structured Text Program (cont'd)

```
                                //Generate Stop2
LOAD_TMP_BIT 4.5                //Global Stop
OR_NOT_TMP_BIT 12.12            //NOT Powerup Done
SET_TMP_BIT 4.7                 //save partial Stop7
LOAD_TMP_BIT 0.0                //PLC Comm Loss from scratch
AND_TMP_BIT 3.0                 //PLC active
AND_NOT_TMP_BIT 4.2             //NOT LO2 PLC fallback value
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2
LOAD_TMP_BIT 0.1                //HMI Comm Loss from scratch
AND_TMP_BIT 3.1                 //HMI active
AND_NOT_TMP_BIT 4.4             //NOT LO1 HMI fallback value
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save partial Stop2

LOAD_NOT_BIT 704.0              //NOT PLC Run1
OR_BIT 704.6                    //Low Speed
AND_TMP_BIT 3.0                 //PLC active
AND_TMP_BIT 4.9                 //Run 2
AND_NOT_TMP_BIT 0.0             //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.7                  //Include partial Stop2
SET_TMP_BIT 4.7                 //save final Stop2

//
```

Structured Text Program (cont'd)

```
// Generate Run1 and Run2 Commands
//
                          //Generate Run 1
                          //PLC mode
LOAD_TMP_BIT 12.1         //Input history
AND_NOT_TMP_BIT 12.11     //NOT Bumpless in Process
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 704.0            //PLC Network Run1
AND_BIT 704.6             //PLC Low Speed
AND_TMP_BIT 12.12         //Power-up Done
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
SET_TMP_BIT 12.1          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.0           //PLC active
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
AND_NOT_TMP_BIT 0.0       //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

                          //HMI mode
LOAD_TMP_BIT 12.3         //Input history
SET_TMP_BIT 12.0          //Save previous history
LOAD_TMP_BIT 13.12        //HMI Run1
SET_TMP_BIT 12.3          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.1           //HMI active
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
AND_NOT_TMP_BIT 0.1       //NOT HMI Comm Loss from scratch
AND_NOT_TMP_BIT 4.12      //Lockout Timer
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1
```

Structured Text Program (cont'd)

```
                          //TS mode
LOAD_TMP_BIT 12.5         //Input history
SET_TMP_BIT 12.0          //Save previous history
LOAD_BIT 457.0            //LI1
SET_TMP_BIT 12.5          //Save new history
AND_NOT_TMP_BIT 12.0      //NOT previous history
AND_TMP_BIT 3.2           //TS active
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
AND_NOT_TMP_BIT 4.12      //Lockout Timer
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run1

                          //PLC Fallback
LOAD_TMP_BIT 4.1          //PLC fallback value
AND_TMP_BIT 3.0           //PLC active
AND_TMP_BIT 0.0           //PLC Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run 1

                          //HMI Fallback
LOAD_TMP_BIT 4.3          //HMI fallback value
AND_TMP_BIT 3.1           //HMI active
AND_TMP_BIT 0.1           //HMI Comm Loss from scratch
OR_TMP_BIT 4.8            //Include previous result
SET_TMP_BIT 4.8           //save partial Run 1

                          //3wire latch
AND_NOT_TMP_BIT 4.6       //NOT Stop 1
AND_NOT_TMP_BIT 4.13      //NOT Swapping
AND_NOT_TMP_BIT 12.7      //NOT Mode Change 1
SET_TMP_BIT 4.8           //save final Run 1
```

Structured Text Program (cont'd)

```
                                //Generate Run 2
                                //PLC mode
LOAD_TMP_BIT 12.2               //Input history
AND_NOT_TMP_BIT 12.11           //NOT Bumpless in Process
SET_TMP_BIT 12.0                //Save previous history
LOAD_BIT 704.0                  //PLC Network Run1
AND_NOT_BIT 704.6               //NOT PLC Low Speed
AND_TMP_BIT 12.12               //Power-up Done
AND_NOT_TMP_BIT 4.7             //NOT Stop 2
SET_TMP_BIT 12.2                //Save new history
AND_NOT_TMP_BIT 12.0            //NOT previous history
AND_TMP_BIT 3.0                 //PLC active
AND_NOT_TMP_BIT 4.7             //NOT Stop2
AND_NOT_TMP_BIT 0.0             //NOT PLC Comm Loss from scratch
OR_TMP_BIT 4.9                  //Include previous result
SET_TMP_BIT 4.9                 //save partial Run2

                                //HMI mode
LOAD_TMP_BIT 12.4               //Input history
SET_TMP_BIT 12.0                //Save previous history
LOAD_TMP_BIT 13.13              //HMI Run2
SET_TMP_BIT 12.4                //Save new history
AND_NOT_TMP_BIT 12.0            //NOT previous history
AND_TMP_BIT 3.1                 //HMI active
AND_NOT_TMP_BIT 4.7             //NOT Stop 2
AND_NOT_TMP_BIT 0.1             //NOT HMI Comm Loss from scratch
AND_NOT_TMP_BIT 4.12            //Lockout Timer
OR_TMP_BIT 4.9                  //Include previous result
SET_TMP_BIT 4.9                 //save partial Run2

                                //TS mode
LOAD_TMP_BIT 12.6               //Input history
SET_TMP_BIT 12.0                //Save previous history
LOAD_BIT 457.1                  //LI2
SET_TMP_BIT 12.6                //Save new history
AND_NOT_TMP_BIT 12.0            //NOT previous history
AND_TMP_BIT 3.2                 //TS active
AND_NOT_TMP_BIT 4.7             //NOT Stop 2
AND_NOT_TMP_BIT 4.12            //Lockout Timer
OR_TMP_BIT 4.9                  //Include previous result
SET_TMP_BIT 4.9                 //save partial Run2
```

Structured Text Program (cont'd)

```
                              //PLC Fallback

LOAD_TMP_BIT 4.2              //PLC fallback value
AND_TMP_BIT 3.0               //PLC active
AND_TMP_BIT 0.0               //PLC Comm Loss from scratch
OR_TMP_BIT 4.9                //Include previous result
SET_TMP_BIT 4.9               //save partial Run2

                              //HMI Fallback
LOAD_TMP_BIT 4.4              //HMI fallback value
AND_TMP_BIT 3.1               //HMI active
AND_TMP_BIT 0.1               //HMI Comm Loss from scratch
OR_TMP_BIT 4.9                //Include previous result
SET_TMP_BIT 4.9               //save partial Run2

                              //3wire latch
AND_NOT_TMP_BIT 4.7           //NOT Stop 2
AND_NOT_TMP_BIT 4.13          //NOT Swapping
AND_NOT_TMP_BIT 12.9          //NOT Mode Change 2
SET_TMP_BIT 4.9               //save final Run 2

//
//Manage Direct Transfer Mechanism
//

//force opposite speed on swap
LOAD_TMP_BIT 4.14             //last speed
AND_TMP_BIT 4.13              //Swapping
OR_TMP_BIT 4.9                //Run 2
SET_TMP_BIT 4.9               //Run 2

//force opposite speed on swap
LOAD_NOT_TMP_BIT 4.14         //NOT last speed
AND_TMP_BIT 4.13              //Swapping
OR_TMP_BIT 4.8                //Run 1
SET_TMP_BIT 4.8               //Run 1
```

Structured Text Program (cont'd)

```
//look for both speeds ON
LOAD_TMP_BIT 4.8         //Run 1
AND_TMP_BIT 4.9          //Run 2
AND_BIT 683.9            //Direct Transfer Enable
AND_NOT_TMP_BIT 3.0      //NOT PLC active
SET_TMP_BIT 4.13         //save Swapping

LOAD_TMP_BIT 3.0         //PLC active
AND_TMP_BIT 4.8          //Run 1
AND_NOT_TMP_BIT 4.14     //NOT last speed
OR_TMP_BIT 4.15          //Two Wire Swap
SET_TMP_BIT 4.15         //save Two Wire Swap

LOAD_TMP_BIT 3.0         //PLC active
AND_TMP_BIT 4.9          //Run 2
AND_TMP_BIT 4.14         //last direction
OR_TMP_BIT 4.15          //Two Wire Swap
SET_TMP_BIT 4.15         //save Two Wire Swap

LOAD_TMP_BIT 3.0         //PLC active
AND_NOT_TMP_BIT 4.8      //Run 1
AND_NOT_TMP_BIT 4.9      //Run 2
OR_TMP_BIT 4.15          //Two Wire Swap
SET_TMP_BIT 4.15         //save Two Wire Swap


//
```

Structured Text Program (cont'd)

```
//Manage Speed Change timer
//
LOAD_K_REG 1            //Low to High Time value (.1s)
LOAD_TMP_BIT 4.10       //Low Speed
ON_SET_TMP_REG 9, 51    //Timer Value - Low to High
LOAD_REG 541            //High to Low Time value
LOAD_TMP_BIT 4.11       //High Speed
OR_NOT_TMP_BIT 12.12    //NOT Power-up Done
ON_SET_TMP_REG 9, 52    //Timer Value - High to Low


LOAD_NOT_TMP_BIT 4.10   //NOT Low Speed
AND_NOT_TMP_BIT 4.11    //NOT High Speed
OR_TMP_BIT 4.13         //swapping
OR_TMP_BIT 11.2         //already timing
OR_TMP_BIT 4.15         //Two Wire Swap
OR_NOT_TMP_BIT 12.12    //NOT Power-up Done
SET_TMP_BIT 11, 0       //Enable Timer

TIMER_TENTHS 9,10,11    //Process lockout timer

//update Swapping flags
LOAD_TMP_BIT 11.0       //Enabled
AND_TMP_BIT 11.2        //timing
SET_TMP_BIT 4.12        //Lockout Timer Active
SET_TMP_BIT 4.15        //Two Wire Swap


//
```

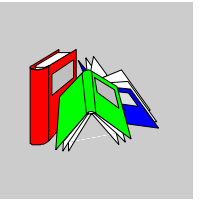
Structured Text Program (cont'd)

```
//Manage Speed 1 and Speed 2 status bits
//

LOAD_NOT_TMP_BIT 4.12   //NOT Lockout Active
OR_TMP_BIT 53.0         //Last Speed-Speed 1
AND_TMP_BIT 4.8         //Run1
AND_NOT_TMP_BIT 4.6     //NOT Stop1
AND_NOT_TMP_BIT 4.11    //NOT High Speed
SET_TMP_BIT 4.10        //save Low Speed
SET_TMP_BIT 53.1        //set last speed=Low

LOAD_NOT_TMP_BIT 4.12   //NOT Lockout Timer Active
OR_NOT_TMP_BIT 53.0     //NOT last Speed-Low
AND_TMP_BIT 4.9         //Run2
AND_NOT_TMP_BIT 4.7     //NOT Stop2
AND_NOT_TMP_BIT 4.10    //NOT Low Speed
SET_TMP_BIT 4.11        //save High Speed
SET_TMP_BIT 53.2        //set last Speed=High

LATCH 53                //last speed latch

LOAD_TMP_BIT 53.0       //Latch value (1=Low Speed)
SET_TMP_BIT 4.14        //save Last Speed

//
// Set Outputs to IMPR
//
                        //Process Output 1
LOAD_TMP_BIT 4.10       //Low Speed
SET_BIT 1200.12         //Output 1
SET_BIT 1200.9          //Aux 1 LED
```

Structured Text Program (cont'd)

```
                              //Process Output 2
LOAD_TMP_BIT 4.11             //High Speed
SET_BIT 1200.13               //Output 2
SET_BIT 1200.10               //Aux 2 LED
SET_BIT 1200.7                //select FLA 2

LOAD_TMP_BIT 4.10             //Low Speed
OR_TMP_BIT 4.11               //High Speed
SET_BIT 1200.0                //Motor Run
SET_NOT_BIT 1200.1            //Motor Stop

LOAD_TMP_BIT 4.12             //Reversing Timer
SET_BIT 1200.4                //Transition Timer

                              //Process other outputs
LOAD_BIT 455.3                //IMPR Alarm status
SET_BIT 1200.14               //Output 3 = Alarm
LOAD_BIT 455.2                //IMPR Fault status
SET_NOT_BIT 1200.15           //Output 4 = Fault
LOAD_BIT 457.4                //Reset Input LI5
SET_BIT 1200.2                //Logic Reset
LOAD_TMP_BIT 3.0              //PLC active
SET_BIT 1200.6                //Logic Local/Remote
LOAD_TMP_BIT 4.6              //Stop 1
OR_TMP_BIT 4.7                //Stop 2
OR_TMP_BIT 4.12               //Lockout Timer Active
SET_BIT 1200.11               //Stop LED
//
// Manage Power-UP Done
//
LOAD_NOT_TMP_BIT 4.12         //Wait for power-up timer
OR_TMP_BIT 12.12              //Latch ON until next power-up
SET_TMP_BIT 12.12             //Power-up Done

// Clear PLC Control on Control Transfer
LOAD_TMP_BIT 4 0              //Control Source Transfer
AND_NOT_BIT 683 10            //NOT Bumpless
LOAD_K_REG 65532              //0xFFFC
AND_REG 704                   //mask off Run1 and Run2
ON_SET_REG 704 54             //Run bits on Bump Control Change
```

Glossary



A

Accumulator Internal register which is used to store the last logic command result. There are 2 different accumulators: the 16-bit accumulator and the 1-bit accumulator.

Argument A number, or an address, representing a value that a program can manipulate in a logic command.

C

Comments Comments are texts you enter to document the purpose of a program. For List programs, enter text on n unnumbered program line. Comments must be inserted after a double slash (//) such as : //COMMENTS GO HERE.

Custom logic register Register which is used to store and provide access to the custom logic program.

D

Device In the broadest terms, any electronic unit that can be added to a network. More specifically, a programmable electronic unit (e.g. PLC, numeric controller or robot)or I/O card.

F

FBD — *Function Block Diagram*. FBD mode allows graphic programming based on the use of predefined function blocks.

FBD editor — A program editor used to create and edit logic programs based on the FBD language. Also called graphical editor.

FLC — *full load current*. Also known as rated current. The current the motor will draw at the rated voltage and rated load. The controller has two FLC settings: FLC1 (Motor Full Load Current Ratio) and FLC2 (Motor High Speed Full Load Current Ratio), each set as a percentage of FLC max.

H

HMI — *Human-Machine Interface*.

L

Logic command — A structured text program consists of a series of logic commands. Each instruction consists of the logic command itself (mnemonic), plus up to 3 arguments.

Logic memory — Part of the internal memory where the custom logic program is stored.

M

Mnemonic — A structured text program consists of a series of logic commands. Each instruction consists of the mnemonic, plus up to 3 arguments.

N

Non-volatile register — Register in the non-volatile memory which can be accessed by logic commands. When power to the controller is cycled, it retains its value settings.

P

PLC — *programmable logic controller*.

S

Structured text editor — A program editor used to create and edit structured text programs.

Structured text language — A program written in instruction structured text language is composed of a series of instructions executed sequentially by the LTM R controller. Each instruction is composed of a line number, a logic command, and an (some) argument(s).

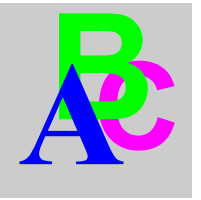
T

Temporary register — Register in the temporary memory which can be accessed by logic commands. When power to the controller is cycled, it does not retain its value settings.

W

Workshop — In the FBD editor, the place where the FBD elements are linked together.

Index



Numerics

1-to1 configuration, 154
1-to-many configuration, 154

A

Accumulator, 43
Add
    Add FBD Function Block, 123
Allocation tables, 175
AND
    AND FBD Function Block, 133
AND_BIT, 59
AND_K, 75
AND_NOT_BIT, 61
AND_NOT_NV_BIT, 62
AND_NOT_TMP_BIT, 61
AND_NV_BIT, 60
AND_NV_REG, 76
AND_REG, 75
AND_TMP_BIT, 60
AND_TMP_REG, 76
Argument
    Grid view, 42
    Text view, 39

C

Comments, 166
    Grid view, 42
    Text view, 39
COMP_K_REG, 72
COMP_NV_REG, 74
COMP_REG, 72
COMP_TMP_REG, 73
Compare
    Compare FBD Function Block, 122
Compiling, 103, 143
Compiling a program, 103
Computation Blocks, 122
Connected, 157
Connection, 154
    Connetion to the LTM R controller, 153
Constant Bit
    Constant Bit FBD Function Block, 125
Constant Word
    Constant Word FBD Function Block, 125
COUNTER, 91
Counter
    Counter FBD Function Block, 129
Counter NV
    Counter NV FBD Function Block, 130
COUNTER_NV, 92
Custom logic editor
    Boolean logic commands, 44
    Commands, 43
    Counter logic commands, 51
    Latch logic commands, 51
    Math logic commands, 52
    Register logic commands, 46
    Timer logic commands, 50
Custom operating modes, 15
Custom pre-defined program, 17

D

Data transfer, 24
Display
    Options, 148
Division
    Division FBD Function Block, 124

E

Error, 106

F

FBD
    Resource management, 142
FBD Blocks
    Deleting, 146
    Duplicating, 146
    Inserting, 138
    Linking, 139
    Properties, 141
    Selecting, 145
FBD Blocks Properties, 141
FBD editor
    Creating an FBD file, 117
    Using the FBD language, 117
FBD Editor Toolbox, 120
FBD Elements
    Computation Blocks, 122
    Function Blocks, 128
    Inputs Blocks, 125
    Logic Blocks, 133
    Outputs Blocks, 134
FBD language, 115, 116
FBD Program
    How to Edit a FBD Program, 119
Full custom program, 17
Function Block Diagram, 115
Function Blocks, 128

G

Grid view, 40

I

Initialization, 156
Inputs Blocks, 125
Inserting FBD Blocks, 138

L

LATCH, 89
LATCH_NV, 90
Linking FBD Blocks, 139
LOAD_BIT, 56
LOAD_K_BIT, 56
LOAD_K_REG, 70
LOAD_NOT_BIT, 58
LOAD_NOT_NV_BIT, 59
LOAD_NOT_TMP_BIT, 58
LOAD_NV_BIT, 57
LOAD_NV_REG, 71
LOAD_REG, 70
LOAD_TMP_BIT, 57
LOAD_TMP_REG, 71
Logic Blocks, 133
Logic command
    Grid view, 41
    Text view, 38
Logic Commands, 43

logic commands
AND_BIT, 59
AND_K, 75
AND_NOT_BIT, 61
AND_NOT_NV_BIT, 62
AND_NOT_TMP_BIT, 61
AND_NV_BIT, 60
AND_NV_REG, 76
AND_REG, 75
AND_TMP_BIT, 60
AND_TMP_REG, 76
COMP_K_REG, 72
COMP_NV_REG, 74
COMP_REG, 72
COMP_TMP_REG, 73
COUNTER, 91
COUNTER_NV, 92
LATCH, 89
LATCH_NV, 90
LOAD_BIT, 56
LOAD_K_BIT, 56
LOAD_K_REG, 70
LOAD_NOT_BIT, 58
LOAD_NOT_NV_BIT, 59
LOAD_NOT_TMP_BIT, 58
LOAD_NV_BIT, 57
LOAD_NV_REG, 71
LOAD_REG, 70
LOAD_TMP_BIT, 57
LOAD_TMP_REG, 71
LOGIC_ID, 54
NOP, 54
ON_ADD, 93
ON_DIV, 96
ON_MUL, 95
ON_SET_NV_REG, 84
ON_SET_REG, 83
ON_SET_TMP_REG, 83
ON_SUB, 94
OR_BIT, 62
OR_K, 77
OR_NOT_BIT, 64
OR_NOT_NV_BIT, 65
OR_NOT_TMP_BIT, 65
OR_NV_BIT, 63
OR_NV_REG, 78
OR_REG, 77
OR_TMP_BIT, 63
OR_TMP_REG, 78
SET_BIT, 66
SET_NOT_BIT, 67
SET_NOT_NV_BIT, 68
SET_NOT_TMP_BIT, 68
SET_NV_BIT, 67
SET_TMP_BIT, 66
TIMER_K_SEC, 87
TIMER_K_TENTHS, 88
TIMER_SEC, 85
TIMER_TENTHS, 86
XOR_K, 79
XOR_NV_REG, 82
XOR_REG, 80
XOR_TMP_REG, 81
Logic Primitive window, 113
Logic Simulator, 112
LOGIC_ID, 54

M

Memory
Logic memory characteristics, 24
Multiplication
Multiplication FBD Function Block, 124
Mux
Mux FBD Function Block, 131

N

Non Volatile Latch
Non Volatile Latch FBD Function Block, 131
NOP, 54
NOT
NOT FBD Function Block, 133

O

ON_ADD, 93
ON_DIV, 96
ON_MUL, 95
ON_SET_NV_REG, 84
ON_SET_REG, 83

ON_SET_TMP_REG, 83
ON_SUB, 94
OR
  OR FBD Function Block, 133
OR_BIT, 62
OR_K, 77
OR_NOT_BIT, 64
OR_NOT_NV_BIT, 65
OR_NOT_TMP_BIT, 65
OR_NV_BIT, 63
OR_NV_REG, 78
OR_REG, 77
OR_TMP_BIT, 63
OR_TMP_REG, 78
Output, 108
Outputs Blocks, 134

P

PCode, 104
Pre-defined operating modes, 15
Program validity, 162
Programming sections, 173
Programming Strategy, 166
Programming strategy, 168, 173, 175

Q

Quick Watch, 110

R

Register
  Non-volatile register, 25
  Register 1200, 27
  Register 1201, 27
  Register 1202, 27
  Register 1203, 27
  Register 1204, 27
  Register 1205, 27
  Registers 1301 to 1399, 25
  Temporary register, 25, 170
  Temporary registers 4 bit 13, 172
  Temporary registers 50 to 54, 172
Register 1200, 113
Register 457, 114
Register 458, 113
Register Bit In
  Register Bit In FBD Function Block, 126
Register Bit Out
  Register Bit Out FBD Function Block, 134
Register NV Bit In
  Register NV Bit In FBD Function Block, 126
Register NV Bit Out
  Register NV Bit Out FBD Function Block, 135
Register NV Word In
  Register NV Word In FBD Function Block, 127
Register NV Word Out
  Register NV Word Out FBD Function Block, 135
Register Temp Bit In
  Register Temp Bit In FBD Function Block, 127
Register Temp Bit Out
  Register Temp Bit Out FBD Function Block, 136
Register Temp Word In
  Register Temp Word In FBD Function Block, 127
Register Temp Word Out
  Register Temp Word Out FBD Function Block, 136
Register View, 113
Register Word In
  Register Word In FBD Function Block, 126
Register Word Out
  Register Word Out FBD Function Block, 135
Resource management, 142

S

Saving, 39
  Logic files, 159
Scan, 168
Selecting FBD Blocks, 145
SET_BIT, 66
SET_NOT_BIT, 67
SET_NOT_NV_BIT, 68
SET_NOT_TMP_BIT, 68
SET_NV_BIT, 67
SET_TMP_BIT, 66
Simulator, 112

Start a Trace, 114
Structured text editor
  Editing a structured text file, 31
  Using the structured text editor, 31
Structured text program, 30
Substraction
  Substraction FBD Function Block, 123

T

TeSys® T
  Motor management system, 12
Text view, 36
Timer Seconds
  Timer Seconds FBD Function Block, 132
Timer TenthSeconds
  Timer TenthSeconds FBD Function Blocks, 132
TIMER_K_SEC, 87
TIMER_K_TENTHS, 88
TIMER_SEC, 85
TIMER_TENTHS, 86
Toolbox
  Computation Blocks, 122
  FBD Editor Toolbox, 120
  Function Blocks, 128
  Inputs Blocks, 125
  Logic Blocks, 133
  Outputs Blocks, 134
Transferring
  Execution, 162
  Logic files, 158
  LTM R controller to PC, 158
  PC to LTM R controller, 160

V

Variables
  Custom logic variables, 25
  LTM R variables, 26
View window, 113
Volatile Latch
  Volatile Latch FBD Function Block, 130

W

Wiring requirements, 168
Workspace
  Appearance options, 149
  Graph options, 149
  Options, 148

X

XOR_K, 79
XOR_NV_REG, 82
XOR_REG, 80
XOR_TMP_REG, 81

You can download this technical publication and other technical information from our website at http://www.telemecanique.com.

Visit http://www.schneider-electric.com for your nearest Schneider Electric affiliate.

© 2006 Schneider Electric. All Rights Reserved.

12/2006